Igor Đurović

TEORIJA INFORMACIJA I KODOVA

IGOR ĐUROVIĆ

TEORIJA INFORMACIJA I KODOVA

Prof. dr Igor Đurović
TEORIJA INFORMACIJA I KODOVA
Prvo izdanje

Izdavač
Univerzitet Crne Gore
Cetinjska br. 2, Podgorica
www.ucg.ac.me

Za izdavača
Prof. dr Vladimir Božović, rektor

Glavni i odgovorni urednik
Prof. dr Stevo Popović

Urednik izdanja
Prof. dr Gojko Joksimović

Recenzije
Prof. dr Predrag Ivaniš
Prof. dr Vojin Šenk
Prof. dr Slobodan Đukanović

Lektura
Irena Stešević

Slog
Božina Bulatović

Tehnički urednik
Ivan Živković

Objavljivanje ove univerzitetske publikacije odobrio je Senat Univerziteta Crne Gore odlukom br. 03-789/1 od 15. aprila 2022. godine.

CIP - Каталогизација у публикацији
Национална библиотека Црне Горе, Цетиње

ISBN 978-86-7664-236-6
COBISS.CG-ID 25216516

IGOR ĐUROVIĆ

TEORIJA INFORMACIJA I KODOVA

PODGORICA, 2023.

Zbog mnogih časova uskraćenih njima, udžbenik posvećujem mojoj porodici Mariji, Ivani i Jeleni.

SADRŽAJ

PREDGOVOR

PREDGOVOR

Poštovani čitaoci, nakon 18 godina, od kada je disciplina Teorija informacija i kodova uvrštena u nastavni plan i program na Elektrotehničkom fakultetu u Podgorici, odlučio sam da za ovaj kurs ponudim materijale u obliku knjige. Razlozi za ovoliko čekanje počivaju na činjenici da na našem jeziku postoje dvije veoma dobre knjige iz ove oblasti, autora Drajića i Ivaniša, te Sinkovića, a dostupne su i prezentacije preko kojih se ova materija izučava na Fakultetu tehničkih nauka u Novom Sadu. Literatura na engleskom jeziku daleko je bogatija pošto se ova disciplina, u manjem ili većem broju kurseva, izučava na gotovo svim univerzitetima svijeta. Mnogi od materijala su dostupni preko interneta, a na istu temu su napisane desetine knjiga, od tutorijalno-teorijskih, preko matematičkih, do praktično-algoritamskih. Mnoge knjige su usko fokusirane, dok druge daju mnogo širi pregled. Stoga, ideja je bila da se, pored knjiga na našem jeziku, oslonimo na predavanja Univerziteta u Juti, SAD, koja su dobro korespondirala sa planom i programom predmeta. Prve godine su napisane pripreme za nastavu, koje su distribuirane studentima. Pretpostavke od kojih se krenulo pokazale su se kao pogrešne. Disciplina se na Elektrotehničkom fakultetu u Podgorici izučavala u V semestru, kada studenti još nisu u dovoljnoj mjeri ovladali telekomunikacijama, a često ni neophodnim matematičkim konceptima. Stoga smo se morali odreći akademske preciznosti i teorema-dokaz pristupa i krenuti pravcem objašnjavanja posljedica matematičkih koncepata i razumijevanja materije. Drugi događaj koji je uticao na promjenu načina izlaganja u okviru našeg kursa bio je uvođenje ove discipline na specijalističke studije primijenjenog računarstva. Bio je veliki izazov da ovu komplikovanu materiju primaknemo studentima primijenjenih studija, što je u mnogo čemu zahtijevalo izmjene i u načinu izlaganja materije, ali, interesantno, ne i u samoj materiji koja je obuhvaćena. Pokazalo se da studenti ne vole elegantne matematičke dokaze koji zahtijevaju poznavanje prethodnih činjenica, već su bili spremni da prihvate ponekad duže i manje elegantne elaboracije koje zahtijevaju manji broj prethodno utvrđenih matematičkih koncepata. Ova knjiga je

nastala petnaestogodišnjim pedagoškim radom i sadrži niz originalnih objašnjenja, kao i brojne primjere koji nisu dostupni u drugim knjigama i materijalima. Neki koncepti su ilustrovani i preko kratkih programskih realizacija. Dio materijala označen je zvjezdicama kako bi se izdvojio materijal namijenjen za studente i istraživače zainteresovane za teorijske teme, kao i one koji su na postdiplomskom nivou odabrali naprednu verziju kursa. Prvih sedam poglavlja odnosi se na materiju koja je vezana za osnovni kurs, dok je preostalo (osmo) poglavlje, s nekim elementima prethodnih poglavlja, dio obaveznog gradiva za napredni kurs ovog predmeta.

Teorija informacija i teorija kodova su oblasti koje su bazirane na dvije različite grupe alata. Tako je teorija informacija zasnovana na teoriji vjerovatnoće i statistike (koja je izuzetno neomiljena kod studenata), dok je savremena kodna teorija zasnovana na teoriji algebarskih grupa. Dakle, jedna oblast je vezana za probabilističke, dok je druga vezana za determinističke koncepte. Knjiga započinje kratkim, rudimentarnim pregledom matematičkih koncepata i drugih pojmova koji su neophodni za praćenje ostatka materijala, uključujući opis (model) sistema za prenos informacija. Drugo poglavlje posvećeno je entropiji kao jednom od najvažnijih koncepata teorije informacije, odnosno mjeri količine informacije. U ovom poglavlju date su osnovne forme entropije, veze između pojedinih veličina, dokazi pojedinih stavova itd. Kada su tvrdnje i teoreme detaljno dokazane, to je prevashodno rađeno ne zbog dokaza kao takvog (opet napominjemo da se nije težilo teorema-dokaz principu), već smo neke dokaze prezentirali kako bi studenti ovladali matematičkim alatom potrebnim za rješavanja problema na koje naiđu. Ovo poglavlje zaključujemo Markovljevim sistemima koji dobro modeluju realne poruke.

Treće poglavlje obrađuje kodiranje izvora. Prvo nastojimo da ubijedimo čitaoca da je kompresija podataka moguća. Na primjeru uvodimo, bez dokaza, asimptotsku ekviparticionu osobinu, a zatim dajemo osnovne pomoćne kodove koji se koriste u kodiranju izvora, kao i za neke druge namjene. Date su definicije osnovnih pojmova potrebnih za razmatranje kodiranja izvora, zatim je uvedena I Šenonova teorema, čime su zaokruženi potrebni teorijski alati da bi se moglo preći na tri najpoznatije klase kodova za kodiranje bez gubitaka, koji su jedino i razmatrani u ovoj knjizi: Hafmenovi (engl. *Huffman*), LZW (engl. *Lempel−Ziv−Welch*) i aritmetički kodovi. Pored ovih, zbog teorijskog i istorijskog značaja, opisan je Šenon−Fanoov (engl. *Shannon−Fano*) postupak kodiranja, kao i nekoliko pomoćnih kodova koji se u praksi koriste i kod kodiranja izvora i kod kodiranja kanala, kao i za druge potrebe.

Četvrto poglavlje razmatra komunikacioni kanal. U najvećoj mogućoj mjeri razmatramo ga lišenog fizikalnosti, odnosno pod pretpostavkom da je fizika komunikacionog problema svedena na sistem uslovnih vjerovatnoća koji korespondira posmatranom kanalu. Ujedno, kombinovanje teorije informacija sa, na

primjer, fizičkim karakteristikama propagacije elektromagnetnih talasa/signala bilo bi prekomplikovano, pa se, stoga, analiza obično svodi na analizu grešaka u kanalu koje se opisuju putem uslovnih vjerovatnoća. Započinjemo osnovnim pojmovima i alatima koji se koriste za opis komunikacionog kanala, od kojih je najvažniji koncept kapaciteta kanala. Izvodimo kapacitet kanala za neke od najpoznatijih tipova kanala koji se koriste u teoriji i sreću u praksi i dajemo osnovne postupke koji se prilikom određivanja kapaciteta upotrebljavaju. Priča o kapacitetu i samom kanalu ne može da bude kompletna ako se ne uvede II Šenonova teorema – teorema o kodiranju kanala. Dajemo formulaciju ove teoreme i govorimo o njenim posljedicama na praktične aspekte kodiranja kanala. Pošto je sam dokaz veoma složen, obilježavamo ga zvjezdicom, kao i neke druge teorijske elemente o kanalu, za naprednije studente ili studente postdiplomskih studija. Ovo poglavlje govori i o nekim sofisticiranijim modelima kanala, kao i o praktičnim aspektima i problemima koji se u procesu kodiranja mogu pojaviti.

Peto i šesto poglavlje bave se blok kodovima, osnovnom klasom kodova za kodiranje kanala. U prvom od ova dva poglavlja uvodimo potrebne pojmove, a zatim diskutujemo binarne i nebinarne kodove za detekciju greške. Nakon toga, na intuitivan način uvodimo neke od najpoznatijih blok kodova: pravougaoni, trougaoni i Hemingov (engl. *Hamming*) s varijantama. Uvodimo sve veličine potrebne za kasnije, kao što su Hemingova distanca, Hemingova težina, minimalno Hemingovo rastojanje, te dajemo geometrijsko tumačenje pojedinih kodova. Šesto poglavlje radi sa blok kodovima na mnogo sistematičniji način. Uvodimo prvo definiciju blok kodova u matričnom obliku, putem kontrolne i generatorske matrice. Zatim govorimo o blokovima interlivera i deinterlivera, koji omogućavaju poboljšavanje karakteristika blok kodova u uslovima praktičnih kanala gdje se može desiti nekoliko uzastopnih grešaka u poruci. Zatim objašnjavamo ograničenja koja postoje vezano za matričnu algebru u teoriji kodova, te na osnovu algebarske kodne teorije (uvedene ovdje više intuitivno nego rigorozno matematički) definišemo blok kodove (odnosno Hemingove kodove) putem polinoma. Sva razmatranja su zatim generalizovana i za nebinarne blok kodove. Ukazujemo, zatim, na problem korekcije više od jedne pogreške, koji prevazilazimo putem BCH (engl. *Bose–Chaudhuri–Hocquenghem*) kodova.

Blok kodovi, premda veoma jednostavno i često zadovoljavajuće rješenje, u teoriji kodova posjeduju bitan nedostatak. Naime, rezultati koji se putem ovih kodova postižu znatno odstupaju od dostižnih granica koje su uspostavljene II Šenonovom teoremom i njenim posljedicama. Stoga su u sedmom poglavlju obrađene dvije klase kodova koji su bliži uslovima koje predviđa kodna teorema: konvolucioni i turbo-kodovi. Konvolucioni kodovi su obrađeni relativno detaljno: objašnjena je terminologija, dati principi konstruisanja/generisanja ovih kodova, kao i tri postupka dekodiranja. Objašnjeni su i neki detalji vezani za dizajn kodera. Kod

turbo-kodova morali smo uvesti principe mekog odlučivanja, osnovnu algebru vezanu za maksimalnu vjerodostojnost (odnos vjerovatnoća, engl. *likelihood*). Stoga smo principe i kodiranja, a posebno dekodiranja turbo-kodova obradili više na primjerima, a manje na eksplicitnom i rigoroznom nivou.

U osmom poglavlju sublimirali smo neke elemente koji se ne mogu izučiti u okviru osnovnog kursa teorije informacija i kodova, za koji je ovaj udžbenik namijenjen. Prvo smo obradili problematiku granica koje se mogu postići u kodnim postupcima, generalizujući unekoliko Hemingovu granicu, koju smo uveli u prethodnim poglavljima. Nakon toga, dali smo CRC (engl. *Cyclic Redundancy Check*) kao naprednu tehniku za detekciju većeg broja pogreški u kodnoj riječi, koja je u širokoj upotrebi. Zatim smo uveli Rid−Mulerove (engl. *Reed−Muller*) kodove, sa ciljem demonstracije kako se na osnovu nekih dobrih kodova mogu uspostaviti drugi kodovi s interesantnim osobinama, praćene Rid−Solomonovim (engl. *Reed−Solomon*) kodovima kao i dalje najšire korišćenom klasom nebinarnih kodova. U ovom dijelu dajemo i nekoliko činjenica vezanih za Vandermondovu matricu, koja značajno pomaže u teorijskoj razradi brojnih kodnih šema. Ukratko smo uveli i LDPC (engl. *Low-Density Parity-Check*) kodove, koji posljednjih godina doživljavaju sve veću popularnost u ovoj oblasti. Poglavlje i samu knjigu završavamo pregledom primjene kodova u nekim komunikacionim standardima.

Svako poglavlje sadrži i niz urađenih primjera, kao i djelove koda u MATLAB®-u, koji mogu da posluže za realizaciju koncepata iz teorije informacija i kodova. U ovom dijelu nismo koristili dostupne funkcije, alate i aplikacije u MATLAB-u, već smo to realizovali primitivno, putem osnovnih konstrukcija koje u ovom programskom alatu postoje. Na kraju knjige smo dali i spisak projekata za samostalni rad, koje su studenti prethodnih godina realizovali u okviru našeg kursa.

Naravno, obrazlaganje ove materije ne može da prođe bez pogreški, posebno kada imamo na umu da se u pojedinim djelovima strogo držimo matematičke preciznosti, dok u drugom dijelu vršimo intuitivna uvođenja pojedinih koncepata. Stoga molimo čitaoce da nam ukažu na sve propuste koji su se u okviru ovakvog materijala javili kako bi bili ispravljeni u budućim izdanjima ove knjige.

Konačno, koristim priliku da najtoplije zahvalim kolegama koje su prethodnih godina pomagale u realizaciji kursa Teorija informacija i kodova, kojem je prevashodno namijenjen ovaj udžbenik, a samim tim su pomogli i prikupljanju, sistematizovanju i ispravljanju ovog materijala: dr Đuru Stojanoviću, mr Predragu Rakoviću, dr Neveni Radović, mr Nikoli Bulatoviću, dr Marku Simeunoviću i dr Slobodanu Đukanoviću.

Podgorica, decembar 2021.

AUTOR

POGLAVLJE I

MODEL SISTEMA ZA PRENOS INFORMACIJA I MATEMATIČKI ALATI

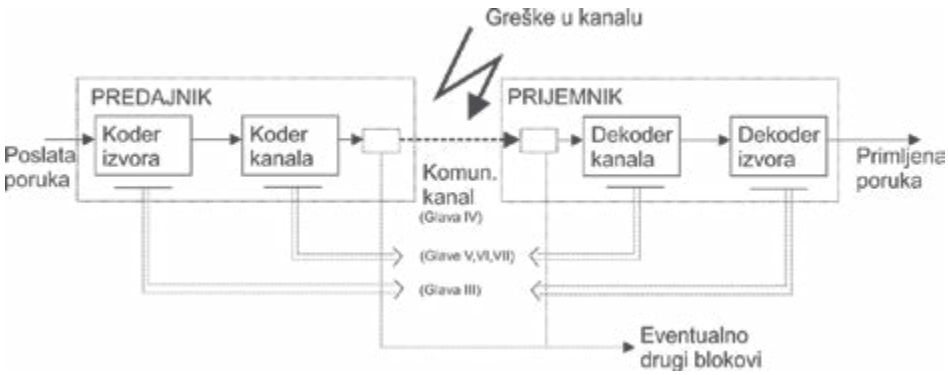
I. MODEL SISTEMA ZA PRENOS INFORMACIJA I MATEMATIČKI ALATI

Poglavlje započinjemo modelom sistema za prenos informacija. Ovaj model se može shvatiti i kao sadržaj ove knjige, gdje ćemo se, uz neke od istorijskih događaja i ličnosti koji su obilježili razvoj ove oblasti, upoznati i s glavnim djelovima sistema za prenos informacija. Kada govorimo o informaciji, vrijedi reći da se ona prikazuje/kodira u obliku signala, pa smo u okviru druge sekcije uveli osnovne tipove signala koji mogu biti nosioci informacija. Potrebno je istaći da će, osim u nekoliko izuzetnih situacija, uvijek biti posmatrani diskretni signali, odnosno informacije koje mogu uzeti diskretan (konačan) broj vrijednosti. Teorija informacija i teorija kodova su različite ne samo po problemima koje razmatraju nego i po osnovnim matematičkim alatima koje koriste. Tako se u teoriji informacija kao osnovni matematički alat koriste elementi teorije vjerovatnoće, jer su po svojoj strukturi informacije slučajni događaji. Stoga, u trećoj sekciji ovoga poglavlja dati su najosnovniji koncepti iz teorije vjerovatnoće kako bi se omogućilo praćenje ostatka ove knjige. Još suptilnije, informacije su po svojim karakteristikama, slučajni procesi. Stoga, u Sekciji I.4 ovoga poglavlja dajemo neophodne definicije slučajnih procesa. Zbog složenosti ove materije, prenos informacija će, u ostatku udžbenika, uglavnom biti sagledavan u smislu vjerovatnoća događaja, a koncepte gdje je potrebno uvesti slučajne procese, posmatraćemo više empirijski. Teorija kodova, za razliku od teorije informacija, potegla je za determinističkim matematičkim alatom – linearnom algebrom. Stoga je peti dio ovog poglavlja posvećen nekim od koncepata linearne algebre koji se koriste u teoriji kodova (polja, grupe, vektorski prostori itd.), ponovo samo u onom obimu koji je neophodan za praćenje ovoga kursa. Na kraju dajemo Euklidski algoritam za određivanje najvećeg zajedničkog djelioca, koji, primijenjen na polinomijalnu algebru sa koeficijentima polinoma iz konačnog polja, ima veliku primjenu u teoriji kodova. Ovo poglavlje, kao i ostala, praćeno je urađenim primjerima, kao i djelovima programskog koda, u svrhu ilustracije uvedenih koncepata.

I.1 Model sistema za prenos informacija

Bez potrebe za matematičkom preciznošću, **podatak** možemo shvatiti kao predstavu ili vrijednost neke osobine objekata. Na primjer, za osobu imamo podatke za: ime (npr. „Ivan“), prezime („Janković“), pol („m“), datum rođenja (12. 11. 1999), boju očiju („zelena“), broj položenih ispita (19), indeks uspjeha (8.23) itd. Dakle, u ovom slučaju, podaci su: „Ivan“, „Janković“, „m“, 12. 11. 1999, „zelena“, 19, 8.23. **Informacija** je korisni dio koji se može dobiti iz podataka. Kako to da se informacija razlikuje od podataka? Razlog je u činjenici da su podaci često **redundantni**, odnosno posjeduju višak. Na primjer „dana 11. 11. 2017, u subotu“ posjeduje redundantan podatak „subota“, jer je 11. 11. 2017. sigurno „subota“. Slično, ako studentu znamo sve ocjene, mi zajedno s njima znamo i prosječnu ocjenu. Tačno je da podatak o prosječnoj ocjeni može učiniti neku tabelu ili bazu podataka preglednijom, ali taj podatak ne donosi novu informaciju, odnosno ne umanjuje naše neznanje o pojavi. **Signali** nosioci informacije (ili podataka) fizičke su predstave kojima zapisujemo ili prenosimo podatke ili informacije. U digitalnom svijetu to su bitovi, odnosno fizičke pojave koje predstavljaju bitove (namagnetisanja, naelektrisanja itd.). To mogu biti izmjereni naponi ili struje, jačina osvjetljaja, jačina zvuka itd. Signali će biti razmatrani kao nizovi izmjerenih veličina, odnosno mi ćemo ih apstrahovati nizovima brojeva.

Jednostavan model sistema za prenos informacija prikazan je na Slici I.1. Pošiljalac informacije (**izvor**) bira signale iz nekoga skupa i njihovim slanjem želi da obavijesti **prijemnik** informacije o nekim činjenicama. Signali nosioci informacije obično uzimaju vrijednosti iz određenog skupa. Za nas će taj skup najčešće biti **diskretan**, odnosno sa konačnim brojem mogućih vrijednosti. Takav skup se obično naziva **alfabetom** i može se označiti kao X:



Slika I.1. Model komunikacionog kanala. Naznačena su i poglavlja ove knjige u kojima se izučavaju pojedini blokovi

$$X = \{x_1, x_2, \ldots, x_N\}.$$

Pohranjivanje velike količine podataka u memoriji ili komuniciranje velikom količinom podataka zahtijeva znatne materijalne izdatke. Stoga je cilj **kodera izvora** da te zahtjeve umanji, kako bi se navedeni troškovi smanjili. Umanjivanje se obavlja uklanjanjem gorepomenute redundancije. Kodiranje i dekodiranje izvora biće razmatrani u trećem poglavlju ove knjige.

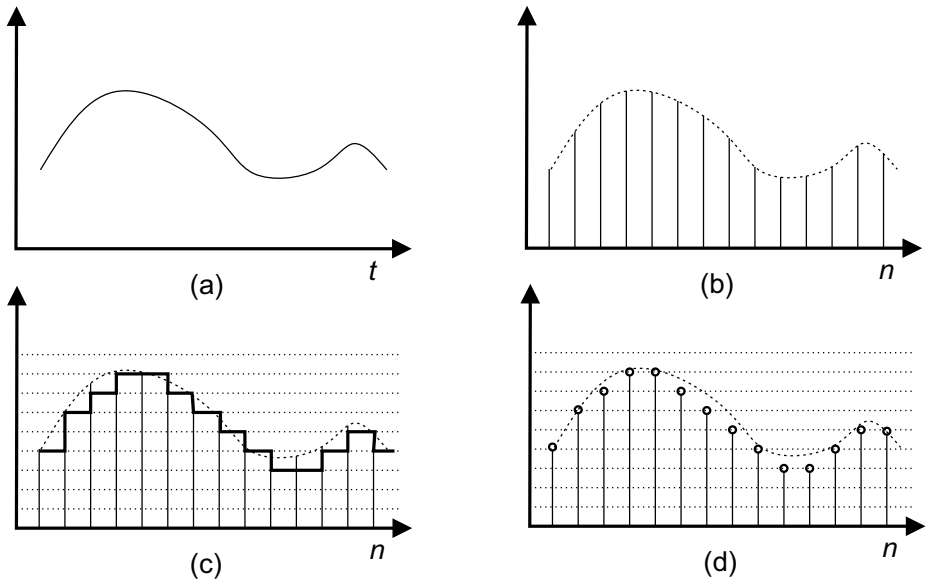
Prilikom prenosa informacija moguće su pojave grešaka u **komunikacionom kanalu**. Taj kanal može biti fizički medij u kojem se ostvaruje radio-veza za prenos telefonskog, radio ili televizijskog signala, računarska mreža, spojni put u obliku žica ili kablova (bakarnih ili optičkih), memorijski mediji (USB, CD, DVD, Blue-ray ili hard-diskovi), voda (za prenos ultrazvuka ili za podvodne sonarne komunikacije), niz usta−uvo u igri gluvih telefona, papir, slika, gramofonska ploča, magnetna traka itd.

Uzroci grešaka u komunikacionim kanalima grubo se mogu podijeliti u: (a) stalno prisutne prirodne slučajne pojave male snage (nazivaju se obično šumovi, a prouzrokovane su toplotnim kretanjem, interakcijom velikog broja objekata itd.); (b) rijetke, ali veoma snažne prirodne pojave (nazivaju se impulsnim šumovima, a mogu biti prouzrokovane grmljavinama, drugim atmosferskim pojavama, lomljenjem leda u podvodnim komunikacijama itd.); (c) ljudske aktivnosti (npr.: rad automobilskih motora, varničenja, rad različitih uređaja u industriji itd.); (d) druge komunikacije (slične komunikacije, po pravilu, negativnije utiču od onih koje se više razlikuju); (e) namjerno prouzrokovane smetnje u cilju onemogućavanja naše komunikacije. Stoga je neophodno komunikaciju zaštititi kodom kanala kako bi se mogle ispraviti greške u prenosu. Na prijemnoj strani mora postojati blok koji se naziva dekoder kanala, koji je u stanju da, uz ispravljanje grešaka koje su se dogodile u kanalu, ispravno dekodira poruku koja je poslata. Komunikacionom kanalu je posvećeno četvrto poglavlje ove knjige, dok se kodiranje kanala obuhvata tri poglavlja: peto, šesto i sedmo (uz neke napredne elemente i u osmom poglavlju). Pored navedenih blokova, u sistemu mogu postojati i drugi, kao što su **interliver** i **deinterliver** i blokovi za kriptografsku zaštitu itd. Interliving poruka će biti obrađena u okviru kodiranja kanala. **Kriptografija**, kao posebno značajna podoblast teorije kodova, zbog svoga obima nije predmet ove knjige.

Može se reći da je tvorac obje ove naučne discipline, i teorije informacija i teorije kodova, američki naučnik Klod Šenon (engl. *Claude Shannon*). On je, u prvim godinama koje su slijedile Drugi svjetski rat, publikovao dva naučna rada u kojima je dao osnove obje discipline. Te osnove se mogu sublimirati u dvije teoreme koje po njemu nose ime i koje uspostavljaju fundamentalne granice o tome što se može postići u procesu kodiranja izvora i kanala. Interesantna je činjenica da Šenon nije bio naročito uspješan u dizajniranju kodova. Međutim, relativno brzo

nakon publikovanja ovih radova, pojavila su se dva koda − Hafmenov (engl. *Huffman*) za kodiranje izvora i Hemingov (engl. *Hamming*) za kodiranje kanala, koji su na efektan način dali praktične upotrebljive metode i za kodiranje kanala i za kodiranje izvora. Šenonov pionirski rad još je više osnažen činjenicom da je bio mentor desetku izuzetnih naučnika, koji su brzo unaprijedili ovu oblast i na taj način uticali na ono što se naziva informatičkom revolucijom.

Brojni izazovi, posebno u teoriji kodova, postoje i danas. Činjenica je da su neki od ranih rezultata bili kreirani pod idealizovanim pretpostavkama o uslovima prenosa informacija. Stoga je relativno dugo bio izazov da se takvi rezultati unaprijede tako da pokriju i uslove realnog prenosa informacija. Rezultati koje je postigao Šenon našli su, gotovo odmah, primjenu u svemirskim istraživanjima i vojnim komunikacijama, ali su mnogo bitnije i revolucionarnije uticali na savremene civilne komunikacije, koje od tada nezaustavljivo napreduju, donoseći gotovo svakodnevno nove mogućnosti. Stoga je značaj rezultata koje je objavio Šenon tako naglašen. Neki misle da je ovo najveće unapređenje u nauci nakon Njutna i Ajnštajna. Naravno, nije Šenon ove rezultate formulisao bez značajne postojeće podloge. Posebno su bile značajne matematičke osnove koje su prethodno dali Kolmogorov, Vitaker (engl. *Whitaker*), Markov i dr., ali je činjenica da je Šenon uspio da ova saznanja objedini, da im dâ jedinstveno tumačenje u smislu konzistentne teorije, koja je, s druge strane, direktno bila povezana s praktičnim potrebama i razvojem u komunikacijama, s poluprovodničkom elektronikom i digitalnim raču-



Slika I.2. Tipovi signala nosilaca informacije: (a) kontinualni, (b) diskretni, (c) kontinualni sa vrijednostima iz konačnog skupa, (d) digitalni

narima. Dodatni multiplikativni impuls bili su naučnici i stručnjaci koje je Šenon oblikovao, prije svega, kroz mentorstvo na doktorskim radovima. Oni su svoj rad nastavili na mnogim univerzitetima, u vladinim agencijama i kompanijama, što je doprinijelo eksplozivnom razvoju ove oblasti, koja sve do danas oblikuje svijet u kome živimo i ima uticaj na sve oblasti ljudskog rada i života.

I.2 Osnovni tipovi signala za prenos informacija

U opštem slučaju, signal koji se koristi za prenos informacija (signal nosilac informacije) može da uzme četiri oblika: analogni (kontinualni), diskretni, kontinualni sa diskretnim vrijednostima amplitude i digitalni. Na Slici I.2 ilustrovani su ovi tipovi signala:

- analogni (kontinualni) signal postoji u svakom vremenskom trenutku datog intervala i može uzimati bilo koju vrijednost iz nekog domena (Slika I.2(a));
- diskretni signal uzima vrijednosti samo u nekim diskretnim pozicijama (odbirci) u vremenu (Slika I.2(b)), ali može uzeti bilo koju vrijednost iz nekog domena;
- analogni signal sa Slike I.2(c) definisan je za bilo koji trenutak datog intervala, ali uzima vrijednosti iz diskretnog skupa;
- digitalni signal je diskretan i po vremenu i po vrijednostima (Slika I.2(d)). Naziva se digitalnim pošto se može jednostavno prikazati u našim računarskim sistemima, putem brojeva ili cifara – engl. *digita*.

Osim u nekoliko specijalnih situacija, u knjizi će biti razmatrani samo signali koji uzimaju vrijednosti iz skupa sa konačnim brojem elemenata.

I.3 Pregled elemenata teorije vjerovatnoće

Već smo rekli da su teorija vjerovatnoće i probabilističko modelovanje pojava osnovni alati u oblasti teorije informacija. Kratko ćemo proći kroz osnovne elemente teorije vjerovatnoće koji će biti korišćeni u knjizi. Napomenimo da ćemo već ovdje uvesti neke pojmove kao što su **signal** i **informacija**. Pretpostavimo da u električnom kolu jedino stanje koje se može pojaviti u nekom čvoru je 5V. To stanje onda nije neodređeno; iako apriori (unaprijed) znamo da je u toj tački napon 5V, mjerenje ne moramo ni obaviti. U rječniku teorije informacija kaže se da ovo stanje ne nosi informaciju.

Ako postoji mogućnost da u pomenutoj tački postoji vrijednost 0V i 5V, postoji potreba da se izvrši mjerenje i da se „ukine neodređenost“ o vrijednosti stanja. Sada vrijednosti 0V možemo pridružiti vrijednost 0 i vrijednosti 5V pridružiti vrijednost

1 u binarnoj aritmetici. Za ovu operaciju se kaže da smo kodirali stanje (poruku) putem simbola **binarnog alfabeta** {0,1}. Posmatrajmo događaj bacanja kocke. Ishod tog događaja nama je unaprijed nepoznat i na osnovu dobijenog rezultata mi dobijamo određenu informaciju. Ta informacija je zasigurno veća od one koju smo dobili nakon što smo se upoznali s ishodom događaja bacanja novčića, jer kod novčića postoji neizvjesnost u izboru samo dvije moguće pojave, dok je kod kocke ta neizvjesnost veća i odnosi se na šest mogućih ishoda.

Dalje, posmatrajmo mogućnost da imamo trideset mogućih naponskih stanja, 0V, 1V, 2V, ... 29V, kojima možemo pridružiti slova naše abecede {A, B, C, ... Ž} ili azbuke {А, Б, В, ... Ш}. Sada zamislimo da nam neko, mijenjajući stanje u tački kola, prenosi poruku. Mi tu poruku možemo tumačiti mjerenjem napona u pojedinim trenucima. Mjereći ovaj napon mi „čitamo“ tekst koji nam je otpremljen. Ova operacija se uslovno može zvati dekodiranje. Da bi se način „prenosa“ informacije optimizovao (poboljšao), neophodno je odrediti prirodu poruke koja se prenosi. Znači, mi očekujemo neku poruku nepoznate sadržine iz skupa svih poruka koje se na našem jeziku mogu izgovoriti. Da bismo tačno kvantifikovali mogućnost pojavljivanja nekog slova, koristimo pojam vjerovatnoće. Tako možemo reći da je vjerovatnoća pojavljivanja slova ‘A’, u ukupnom skupu slova u našem jeziku, neka vrijednost. Očigledno je ta vrijednost veća nego za pojavljivanje slova ‘Š’. Da bismo pravilno optimizovali način mjerenja (odnosno prenosa informacija), trebalo bi da nam mjerna skala oko vrijednosti napona, koja daje slovo ‘A’, bude preciznija (osjetljivija) nego ona oko slova ‘Š’ (možda i obrnuto, o čemu će više biti riječi u narednom poglavlju). Isto se dešava kod prenosa informacija. Dakle, poznavanje vjerovatnoće pojavljivanja nekog slova u poruci opredjeljuje performanse sistema. Sistem koji govori o porukama koje su unaprijed nepoznate sadržine je **stohastički** ili **slučajan**. Suprotno od ovoga su **deterministički sistemi**. Deterministički sistemi ne nose informaciju, ali su lakši za analizu, pa se stoga često oni koriste za modelovanje pojedinih pojava, posebno u komunikacijama. Teorija informacija, međutim, polazi od slučajnih poruka s određenom vjerovatnoćom pojavljivanja svakog simbola.

Drugi, veoma značajan, razlog za uvođenje pojmova teorije vjerovatnoće u teoriju informacija je modelovanje smetnji koje se mogu dogoditi pri akviziciji, prenosu, prijemu i razumijevanju poruke. Teorijski bi se ove smetnje mogle opisati determinističkim zakonima. Na primjer, kada bismo znali sve parametre prilikom bacanja kocke, kao i sve podatke o okruženju u kojem kocka pada, dobili bismo tačan rezultat bacanja. Međutim, takva analiza je, po pravilu, nemoguća i za nas su dovoljne procjene procesa koji do ishoda dovode. Takvi će procesi biti nazivane šumovima. Važno je napomenuti da neke vrste šumova potiču od termičkog kretanja elektrona, druge su izazvane atmosferskim procesima ili ljudskim aktivnostima. Međutim, mi u ovoj knjizi nećemo razmatrati šum na fizičkom nivou, već samo kao model.

Taj model je dodatno sveden na vjerovatnoće koje opisuju kako fizičke pojave utiču na sistem za prenos informacija.

I.3.1 Procjena vjerovatnoće

Posmatrajmo sada slučajni događaj odabira jednog elementa iz konačnog skupa (alfabeta) **X** koji se sastoji od elemenata $x_i \in \mathbf{X}$, $i = 1,2,...,N$. Vrši se veliki broj „izbora" iz skupa **X**. Neka je svaki x_i element „izvučen" k_i puta. Tada se može reći da je procjena vjerovatnoće pojavljivanja nekog elementa iz skupa **X** jednaka:

$$\hat{p} = \frac{k_i}{k_1 + k_2 + ... + k_N} = \frac{k_i}{\sum_{i=1}^{N} k_i} = \frac{k_i}{K} \qquad \sum_{i=1}^{N} \hat{p}_i = 1$$

gdje je: $k_1 + k_2 + ... + k_N = K$. Ako je K veoma veliki broj (teži beskonačno), nije više riječ o procjeni vjerovatnoće pojavljivanja nekog elementa, već o vjerovatnoći. Uvijek je sporno koliko je dobra procjena vjerovatnoće događaja. Naime, za skup slova našeg jezika potpuno valjana procjena bi obuhvatila sve što je ikada rečeno i napisano na njemu. Čak nam ni takva procjena ne može govoriti o kvalitetu mjerenja (određivanja neke poruke), jer, na primjer, u poruci može biti izostavljen neki često ponavljani karakter. Postoji knjiga na engleskom jeziku u kojoj je izostavljen karakter 'E', inače najčešći karakter u ovom jeziku (knjiga „Gadsby", koju je napisao Ernest Vinsent Rajt, engl. *Ernest Vincet Wright*, 1939. godine). Ako, na primjer, kao osnovu za procjenu vjerovatnoće koristite knjigu kao što je „Tom Sojer", za očekivati je da će broj pojavljivanja karaktera koji se nalaze u imenima junaka biti znatno veći nego što je broj pojava ovih karaktera u standardnom tekstu.

Ako se za trenutak vratimo na naš jezik i pogledamo pojavljivanje pojedinih karaktera, činjenica je da se karakter 'A' pojavljuje više od 10% puta, dok se, na primjer, karakter (glas) 'Dž' ('Џ') pojavljuje sa vjerovatnoćom koja je reda 10^{-5}. Osim u specifičnim tekstovima, vjerovatnoća pojavljivanja ovoga glasa ne prelazi 10^{-4}. Ako uzmemo kratak tekst, postoji značajna vjerovatnoća da se ovaj glas neće pojaviti nijednom, dok ako uzmemo neki specifičan, postoji mogućnost da se ovaj karakter pojavljuje neočekivan broj puta. Stoga je dobra mjera uzeti da se najrjeđi simbol od značaja pojavi barem desetak puta, odnosno da je broj elemenata u skupu koji je korišćen za procjenu vjerovatnoće minimalno $10/p_{\min}$, gdje je $p_{\min}$ najmanja vjerovatnoća simbola iz alfabeta (koji se pojavljuje u porukama). Dakle, u našem slučaju riječ je o skupu od barem 100 hiljada karaktera (u tekstovima koji su statistički korektno odabrani). Kasnije će biti rečeno nešto o veličini alfabeta u složenijim slučajevima.

I.3.2 Združene i uslovne vjerovatnoće

U teoriji vjerovatnoće postoje i kombinovane ili **združene slučajne** promjenljive koje predstavljaju mogućnost da se dogode dva ili više događaja − na primjer, bacanje dva novčića ili kocke. Drugi bitan pojam koji će biti obrađen u ovoj sekciji je uslovna vjerovatnoća. Znamo da se dogodio neki događaj, na primjer, slanje slova 'S' u našem jeziku. Veoma često se iza ovog slova pojavljuju slova 'A', 'E' ili 'T', ali veoma rijetko 'B' ili 'Č', ili ako znamo da je na početku neke riječi bio string „SUD", bitno smo ograničeni – uslovljeni u smislu koja je to riječ koja započinje ovim nizom karaktera. Ovo nas vodi do koncepta **uslovne vjerovatnoće**.

Vjerovatnoću da su se desila dva događaja označavaćemo kao:

$$P(\text{dogodili se } A \text{ i } B) = P(AB).$$

Uslovni događaj predstavlja mogućnost da se neki događaj A dogodi nakon što smo u saznanju da se dogodio događaj B. Uslovna vjerovatnoća se označava kao $P(A \mid B)$. Ova vjerovatnoća se može sračunati iz združene i vjerovatnoće pojedinačnog događaja B:

$$P(A \mid B) = \frac{P(AB)}{P(B)}.$$

Dakle, vidimo da se združena vjerovatnoća može sračunati iz uslovne, kao:

$$P(AB) = P(B)P(A \mid B).$$

Uslovne vjerovatnoće se mogu posmatrati u dva pravca. Znamo da se dogodio neki događaj i pitamo se šta će se dogoditi nakon toga. Međutim, možemo znati ishod i da se pitamo što ga je uzrokovalo, npr.: primili smo neku poruku i pitamo se koja je poruka poslata i koja je u prenosu možda izmijenjena. Dakle, važi:

$$P(AB) = P(B)P(A \mid B) = P(A)P(B \mid A).$$

U slučaju da su dva događaja nezavisna, odnosno da događaj B ne govori ništa o događaju A, onda je združena vjerovatnoća jednaka proizvodu vjerovatnoća događaja A i B:

$$P(AB) = P(A)P(B).$$

Pretpostavimo sljedeću situaciju. Na jednom kraju komunikacionog kanala nalazi se izvor koji emituje skup simbola B_i, $i = 1,..., N$. Pitamo se: koja je vjerovatnoća da se na izlazu iz sistema pojavi simbol A? Očigledno se ovaj simbol može pojaviti kada je poslato B_1 (s vjerovatnoćom $P(B_1)$) koje je postalo A u komunikacionom kanalu (uslovna vjerovatnoća $P(A|B_1)$), pa zatim, ako je poslato B_2 (s vjerovatno-

ćom $P(B_2)$) koje je postalo A u komunikacionom kanalu (uslovna vjerovatnoća $P(A|B_2)$) itd., što se može zapisati kao:

$$P(A) = \sum_{i=1}^{N} P(B_i)P(A \mid B_i).$$

Predmetna relacija često se naziva formulom **totalne vjerovatnoće**. Sada se možemo pitati i u obrnutom smjeru: ako se dogodilo A na prijemu, s kojom vjerovatnoćom je ono što je poslato B_j:

$$P(B_j \mid A) = \frac{P(AB_j)}{P(A)} = \frac{P(A \mid B_j)P(B_j)}{\sum_{i=1}^{N} P(B_i)P(A \mid B_i)}.$$

Predmetna relacija naziva se Bajesovom (ponekad **Bajesovo pravilo**) i opisuje **aposteriori vjerovatnoću** (vjerovatnoću da ako znamo da se dogodio događaj A da ga je uzrokovao događaj B_j). Ilustrujmo neke od izvedenih relacija na dva jednostavna primjera.

Primjer I.1. Pretpostavimo da raspolažemo sa dva novčića od po 1€. Jedan novčić je „fer“, gdje se s vjerovatnoćom 0.5 dešavaju oba ishoda: pismo (vrijednost 1) i glava (vrijednost 0). Pored vas, u igri učestvuje igrač sa novčićem od 1€ koji nije „fer“, već s vjerovatnoćom 0.7 pada na pismo (vrijednost 1), a s vjerovatnoćom 0.3 pada na glavu (vrijednost 0). Posmatrajte ishod C, koji predstavlja sumu dobijenu bacanjem ova dva novčića, i odredite odgovarajuće uslovne i združene vjerovatnoće.

Rješenje: Neka je A ishod bacanja prvog novčića s vrijednostima u alfabetu $A = \{0, 1\}$ s pridruženim vjerovatnoćama $\{0.5, 0.5\}$, to jest $P(A=0) = 0.5$ i $P(A=1) = 0.5$. Slično je kod drugog događaja B, koji je definisan preko istog binarnog alfabeta: $P(B=0) = 0.3$ i $P(B=1) = 0.7$. Pošto su događaji bacanja dva novčića nezavisni, to važi da je vjerovatnoća združenog događaja jednaka proizvodu vjerovatnoća:

$$P(A=0, B=0) = P(A=0)P(B=0) = 0.15,$$
$$P(A=0, B=1) = P(A=0)P(B=1) = 0.35,$$
$$P(A=1, B=0) = P(A=1)P(B=0) = 0.15,$$
$$P(A=1, B=1) = P(A=1)P(B=1) = 0.35.$$

Uvedimo sada događaj C, koji predstavlja sumu ishoda dobijenih bacanjem dva novčića. Alfabet koji odgovara ovom slučaju je $C = \{0, 1, 2\}$. Vjerovatnoće ova tri događaja su: $\{0.15, 0.5, 0{,}35\}$. Sada smo u potpunosti u stanju da uvedemo koncept uslovne vjerovatnoće. Pretpostavimo da smo bacili prvi novčić i da znamo njegov ishod: interesuje nas kolika je vjerovatnoća da dobijemo pojedini ishod

događaja C. Recimo $P(C=2|A=1)$ je jednako $P(\mathrm{C}=2,A=1)/P(A=1)$, odnosno ovo je identično vjerovatnoći $P(B=1)$ da na drugom novčiću imamo vrijednost 1, dok je $P(C=2|A=0)=0$ jer je nemoguć događaj da se samo s jednim novčićem dobije ishod 2. Pregled odgovarajućih uslovnih vjerovatnoća:

$$P(C=2|A=1)=P(C=2,\,A=1)/P(A=1)=P(B=1)=0.7,$$
$$P(C=2|A=0)\ =P(\mathrm{C}=2,\,A=0)/P(A=0)=0,$$
$$P(C=1|A=1)=P(C=1,\,A=1)/P(A=1)=P(B=0)=0.3,$$
$$P(C=1|A=0)=\ P(C=1,\,A=0)/P(A=0)=P(B=1)=0.7,$$
$$P(C=0|A=1)=P(C=0,\,A=1)/P(A=1)=0,$$
$$P(C=0|A=0)=\ P(C=0,\,A=0)/\ P(A=0)=P(B=0)=0.3.$$

Predmetne rezultate je zgodno prikazati tabelarno (Tabela I.1). Iz tabele združenih vjerovatnoća vidimo važnu činjenicu da ako sumiramo združenu vjerovatnoću $P(C,A)$ po jednoj od dvije veličine (recimo, po A), u tom redu ili koloni dobijamo vjerovatnoću druge slučajne promjenljive (u ovom slučaju C). Zbog toga što se ove vjerovatnoće pišu na margini tabele ponekad se nazivaju **marginalnim**. Tabelu uslovnih vjerovatnoća $P(C|A)$ smo dobili tako što je podijeljena svaka vrijednost iz tabele združenih vjerovatnoća s marginalnom vjerovatnoćom koja je upisana u koloni sa marginalnom $P(A)$. Potpuno analogno prethodnom možemo da sračunamo uslovnu aposteriori vjerovatnoću $P(A|C)$ (aposteriori vjerovatnoća nam kaže koja je vjerovatnoća polaznog događaja ako znamo ishod) koja nam kaže kolika je vjerovatnoća događaja A ako nam je poznat ishod događaja C. To je često slučaj koji imamo u komunikacijama i teoriji informacija. Primili smo neki podatak i želimo da odredimo da li je poslat taj ili neki drugi podatak (simbol).

$P(C,A)$	$A=0$	$A=1$	$P(C)$
$C=0$	0.15	0.00	0.15
$C=1$	0.35	0.15	0.50
$C=2$	0.00	0.35	0.35
$P(A)$	0.50	0.5	1.00

$P(C\|A)$	$A=0$	$A=1$
$C=0$	0.3	0
$C=1$	0.7	0.3
$C=2$	0	0.7

Tabela I.1. Združene i marginalne vjerovatnoće za događaje A i C

Provjerimo sada u posmatranom primjeru tačnost sljedeće relacije:

$$P(A)=\sum_{i=1}^{N}P(C_i)P(A\,|\,C_i)$$

$$\begin{aligned}P(A=0)&=P(C=0)P(A=0\,|\,C=0)+P(C=1)P(A=0\,|\,C=1)\\&\quad+P(C=2)P(A=0\,|\,C=2)=\\&=0.15\cdot 1+0.5\cdot 0.7+0.35\cdot 0=0.5\\P(A=1)&=P(C=0)P(A=1\,|\,C=0)+P(C=1)P(A=1\,|\,C=1)\\&\quad+P(C=2)P(A=1\,|\,C=2)=\\&=0.15\cdot 0+0.5\cdot 0.3+0.35\cdot 1=0.5.\end{aligned}$$

Uočimo da su članovi $P(C=i)P(A=j\,|\,C=i)$ zapravo združene vjerovatnoće $P(A=j,C=i)$. Ujedno, ovim je posredno potvrđeno i Bajesovo pravilo:

$$P(C=j\,|\,A=i)=\frac{P(A=i\,|\,C=j)P(C=j)}{\sum_{l=1}^{N}P(C=l)P(A=i\,|\,C=l)}.$$

Provjerimo ga za ovu priliku samo za A=1 i C=1:

$$\begin{aligned}P(C=1\,|\,A=1)&=\frac{P(A=1\,|\,C=1)P(C=1)}{\sum_{i=0}^{2}P(C=i)P(A=1\,|\,C=i)}=\\&=\frac{0.3\cdot 0.5}{0.15\cdot 0+0.3\cdot 0.5+0.35\cdot 1}=\frac{0.15}{0.5}=0.3.\end{aligned}$$ □

Primjer I.2. Bacaju se dvije kocke. Događaj X je suma vrijednosti dobijena bacanjem, dok je događaj Y maksimalna vrijednost dobijena bacanjem. Sračunati uslovnu vjerovatnoću $P(x|y)$.

Rješenje: Tabela I.2 prikazuje združene vjerovatnoće pojedinih događaja. Pojasnimo na jednom primjeru. Suma $X=4$ mogla se dogoditi u slučajevima kada je na prvoj kocki dobijeno 1, a na drugoj 3 i kada je na prvoj kocki dobijeno 3 i na drugoj 1. Odnosno $P(X=4,Y=3)=2/36$ (maksimum je u oba slučaja 3) zato što je vjerovatnoća oba ishoda $\{1,3\}$ i $\{3,1\}$ jednaka $(1/6)\times(1/6)$. Druga situacija za sumu $X=4$ je kada na obje kocke dobijemo ishod 2 (maksimum je $Y=2$), a to se dešava s vjerovatnoćom od $P(X=4,Y=2)=1/36$. Vjerovatnoća nemogućih događaja jednaka je 0. Tako, na primjer, suma ne može biti $X=3$, a maksimum $Y=1$. Na sličan način, u slučaju pojave sigurnog događaja dobijamo vjerovatnoću 1. Relativno jednostavno mogu se sračunati uslovne vjerovatnoće na osnovu ove tabele, dijeljenjem vrijednosti s odgovarajućim marginalnim vjerovatnoćama.

$P(x,y)$	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6	$P(x)$
X=2	1 / 36	0	0	0	0	0	1 / 36
X=3	0	2 / 36	0	0	0	0	2 / 36
X=4	0	1 / 36	2 / 36	0	0	0	3 / 36
X=5	0	0	2 / 36	2 / 36	0	0	4 / 36
X=6	0	0	1 / 36	2 / 36	2 / 36	0	5 / 36
X=7	0	0	0	2 / 36	2 / 36	2 / 36	6 / 36
X=8	0	0	0	1 / 36	2 / 36	2 / 36	5 / 36
X=9	0	0	0	0	2 / 36	2 / 36	4 / 36
X=10	0	0	0	0	1 / 36	2 / 36	3 / 36
X=11	0	0	0	0	0	2 / 36	2 / 36
X=12	0	0	0	0	0	1 / 36	1 / 36
$P(y)$	1/36	3/36	5/36	7/36	9/36	11/36	

Tabela I.2. Združene i marginalne vjerovatnoće za događaje vezane za bacanje dvije kocke

$P(x\|y)$	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6
X=2	1	0	0	0	0	0
X=3	0	2 / 3	0	0	0	0
X=4	0	1 / 3	2 / 5	0	0	0
X=5	0	0	2 / 5	2 / 7	0	0
X=6	0	0	1 / 5	2 / 7	2 / 9	0
X=7	0	0	0	2 / 7	2 / 9	2 / 11
X=8	0	0	0	1 / 7	2 / 9	2 / 11
X=9	0	0	0	0	2 / 9	2 / 11
X=10	0	0	0	0	1 / 9	2 / 11
X=11	0	0	0	0	0	2 / 11
X=12	0	0	0	0	0	1 / 11

Tabela I.3. Uslovne vjerovatnoće $P(x|y)$ za slučaj bacanja dvije kocke

Tabela I.3 prikazuje uslovnu vjerovatnoću $P(x|y)$, dobijenu dijeljenjem ćelija tabele združenih vjerovatnoća s marginalnom $P(y)$. Sami sračunajte uslovnu vjerovatnoću $P(y|x)$ putem istog postupka.

Predmetni primjer biće iskorišćen u narednom poglavlju da bismo njime ilustrovali tačnost relacija između fundamentalnih veličina teorije informacija.□

I.3.3 Kontinualne slučajne promjenljive

Broj simbola nekog alfabeta je obično konačan ma kako bio veliki. Simboli se prenose putem komunikacionog kanala i predstavljeni su nekim fizičkim veličinama. Na primjer, naponom 0V za simbol 'A', 1V za simbol 'B' i dalje cjelobrojnim vrijednostima naponskih nivoa. Zbog djelovanja različitih faktora u kanalu, primljeni signal na prijemu nije cijeli broj. Razlog može biti u slabljenju signala u komunikacionom kanalu, ali i slučajnim pojavama (šumovima) koje ometaju komunikaciju. Ovakve pojave ne mogu da se ograniče samo na diskretni – konačni broj elemenata, već uzimaju vrijednosti u nekom domenu, ali iz skupa od beskonačno mnogo elemenata. Tako smetnja može da ima vrijednost 0.1V, ali može da ima i vrijednost 0.100001V ili 0.000000000001V. Ovakve pojave se karakterišu (nazivaju) **kontinualnim slučajnim promjenljivima**. Premda ćemo u ovoj knjizi pokušati da svedemo rad s ovakvim promjenljivima na najmanju moguću mjeru ipak se radi o značajnom konceptu s kojim ćemo se sudariti nekoliko puta u ovoj knjizi.

Očigledno je ovdje besmisleno govoriti o vjerovatnoći da neka slučajna promjenljiva uzme tačno određenu vrijednost iz datog skupa. Stoga se uvodi **funkcija raspodjele** kao vjerovatnoća da neka slučajna promjenljiva X uzme vrijednost koja je manja ili jednaka nekoj vrijednosti:

$$P(\xi) = P(X \le \xi).$$

Očigledno je da važi:

$$P(-\infty) = 0 \qquad P(\infty) = 1.$$

Ako je slučajna promjenljiva X definisana u domenu $X \in [a,b]$, tada važi:

$$P(\xi) = 0 \quad \xi < a$$
$$P(\xi) = 1 \quad \xi > b.$$

Funkcija raspodjele je neopadajuća veličina:

$$P(\xi_1) \le P(\xi_2) \text{ za } \xi_2 > \xi_1.$$

Ova funkcija ne predstavlja analogiju vjerovatnoćama simbola diskretnog skupa (alfabeta), već tu ulogu igra **funkcija gustine raspodjele**:

$$p(\xi) = P'(\xi) = \frac{dP(\xi)}{d\xi}.$$

Funkcija gustine raspodjele zadovoljava sljedeće osobine i veze sa funkcijom raspodjele:

$$P(\xi) = \int_{-\infty}^{\xi} p(x)dx$$

$$\int_{-\infty}^{\infty} p(x)dx = 1.$$

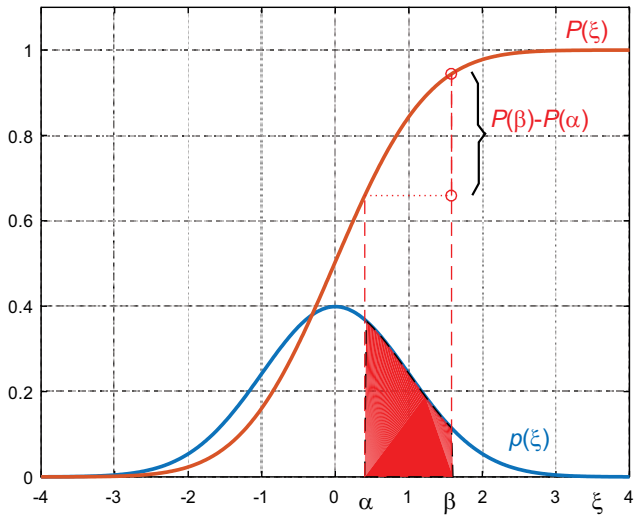
$$\text{Vjerovatnoća}[\xi \in [\alpha,\beta]] = \Pr[\xi \in [\alpha,\beta]] = P(\beta) - P(\alpha) = \int_{\alpha}^{\beta} p(x)dx.$$

Na Slici I.3 prikazana je ilustracija veze između funkcije raspodjele i funkcije gustine raspodjele slučajne promjenljive. Prikazana je i vjerovatnoća da slučajna promjenljiva uzme vrijednost iz datog intervala i to preko integrala funkcije gustine raspodjele i preko razlike funkcije raspodjele.

Kontinualne slučajne promjenljive često se karakterišu putem **srednje vrijednosti** μ i **varijanse** σ^2 (σ se naziva standardna devijacija), koje se mogu izraziti putem funkcije gustine raspodjele na sljedeći način:

$$\mu = \mathrm{E}\{x\} = \int_{-\infty}^{\infty} xp(x)dx \qquad \sigma^2 = \mathrm{E}\{(x-\mu)^2\} = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx.$$

Operator $\mathrm{E}\{\}$ predstavlja matematičko očekivanje. Matematičko očekivanje ukazuje na srednju vrijednost slučajne promjenljive dobijene u velikom broju pokušaja, a varijansa ukazuje na variranje slučajne promjenljive oko srednje vrijednosti. Dvije najpoznatije (ujedno i najjednostavnije) raspodjele slučajnih promjenljivih su:



Slika I.3. Ilustracija funkcije raspodjele i funkcije gustine raspodjele i računanja vjerovatnoće slučajne promjenljive u datim granicama

– uniformna raspodjela

$$p(\xi)=\begin{cases}\dfrac{1}{b-a} & \xi\in[a,b]\\ 0 & \text{drugdje}\end{cases}\qquad P(\xi)=\begin{cases}0 & \xi<a\\ (\xi-a)/(b-a) & \xi\in[a,b]\\ 1 & \xi>b,\end{cases}$$

sa srednjom vrijednošću $\mu=(b+a)/2$ i varijansom $\sigma^2=(b-a)^2/12$;

– Gausova (normalna) raspodjela sa srednjom vrijednošću μ i varijansom σ^2, koja se često označava i kao $\mathbf{N}(\mu,\sigma^2)$:

$$p(\xi)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\xi-\mu)^2}{2\xi^2}\right).$$

Srednju vrijednost i varijansu kod diskretnog alfabeta ima smisla definisati samo onda kada se simboli alfabeta mogu kvantifikovati, odnosno kada je riječ o brojevima ili kada se svakom simbolu alfabeta x_i može pridružiti smislena brojna vrijednost $f(x_i)$. U tom slučaju se srednja vrijednost i varijansa ovakve slučajne promjenljive određuju sumiranjem:

$$\mu=\sum_{i=1}^{N}f(x_i)p(x_i)\qquad\qquad \sigma^2=\sum_{i=1}^{N}(f(x_i)-\mu)^2\,p(x_i).$$

Združena (ponekad se kaže dvodimenziona) funkcija raspodjele $P(\xi,\zeta)$ i funkcija gustine raspodjele $p(\xi,\zeta)$ vezane su međusobno i sa marginalnim veličinama putem sljedećih relacija:

$$p(\xi,\zeta)=\frac{\partial^2P(\xi,\zeta)}{\partial\xi\partial\zeta}\qquad\qquad P(\xi,\zeta)=\int_{-\infty}^{\xi}\int_{-\infty}^{\zeta}p(x,y)dxdy$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}p(x,y)dxdy=1$$

$$\int_{-\infty}^{\infty}p(x,y)dx=p(y)\qquad\qquad \int_{-\infty}^{\infty}p(x,y)dy=p(x).$$

Veza s uslovnim vjerovatnoćama može se uspostaviti na sličan način kao kod diskretnih slučajnih promjenljivih:

$$p(\xi,\zeta)=p(\xi)p(\zeta\,|\,\xi)=p(\zeta)p(\xi\,|\,\zeta).$$

I.4 Slučajni procesi*

Prethodni pregled elemenata teorije vjerovatnoće polazio je od pretpostavke da se statističke karakteristike slučajnih promjenljivih ne mijenjaju tokom posmatranja. U našem slučaju, te statističke karakteristike sveli smo samo na srednju vrijednost

i varijansu, ali ih i u teoriji i u praksi ima više. Ako imamo slučaj promjene karakteristika slučajne promjenljive tokom posmatranja, govorimo o **slučajnim** ili **stohastičkim procesima**. Kod ovakvih pojava očigledno je da imamo dodatne uticaje, pored vjerovatnoća, koji karakterizuju posmatrane promjenljive. Najjednostavnije je uvesti trenutak posmatranja (vrijeme) t kao dodatni faktor koji utiče na pojedinu pojavu. Još je u ranim komunikacijama uočeno da jednog dana imamo dobar kvalitet u komuniciranju, dok drugoga dana ili samo nekoliko sati kasnije kvalitet u komuniciranju može biti znatno lošiji. Više je nego očigledno da su gotovo sve pojave u komunikacijama, pa sljedstveno tome i u teoriji informacija i kodova, slučajni procesi, te da se samo radi pojednostavljivanja razmatranja poseže za modelima kod kojih vjerovatnoće određenih pojava ili statističke karakteristike procesa ne zavise od vremena.

Ponekad se umjesto razmatranja pojedinačnih ishoda, kod slučajnih procesa razmatraju funkcije koje zavise od vremena. Svi mogući ishodi (odnosno, sve moguće funkcije slučajnih ishoda) koji zavise od vremena posmatranja događaja (ili neke druge vremenu ekvivalentne pojave) nazivaju se **ansambl**. Pojedinačne funkcije se nazivaju **realizacijama** ili članovima ansambla.

Da bi se približio način opisivanja osnovnih pojmova i tipova slučajnih procesa, pretpostavimo da imamo funkciju gustine raspodjele ovakve pojave kod koje sada uvodimo eksplicitno zavisnost od trenutka posmatranja $p(\xi,t)$ (na isti način se može prikazati i funkcija raspodjele $P(\xi,t)$). Srednja vrijednost i varijansa ovakvih procesa su takođe funkcije vremena, pa se mogu označiti kao $\mu(t)$ i $\sigma^2(t)$. Slučajni proces se naziva **stacionarnim** kada su funkcije gustine raspodjele (i funkcije raspodjele) jednake $p(\xi,t_1)=p(\xi,t_2)$ (odnosno $P(\xi,t_1)=P(\xi,t_2)$) za svako t_1, t_2. Procesi su **stacionarni u širem smislu** kada im se srednja vrijednost i varijansa ne mijenjaju tokom vremena $\mu(t_1)=\mu(t_2)=\mu$ i $\sigma^2(t_1)=\sigma^2(t_2)=\sigma^2$. Očigledno je svaki stacionarni proces ujedno stacionaran u širem smislu, dok obrnuto ne mora da važi.

Pored statistika (srednje vrijednosti i varijanse) koje su računate za dati trenutak preko svih mogućih realizacija datog slučajnog procesa (preko ansambla), statistike se mogu računati i preko vremena, usrednjavajući sve ishode slučajnog procesa u susjednim vremenskim trenucima. Pretpostavimo da posmatramo neki slučajni proces $X(t)$ koji se može opisati odgovarajućim funkcijama raspodjele i gustine raspodjele ($p(\xi,t)$ i $P(\xi,t)$) i kod kojeg se za svaki trenutak može odrediti srednja vrijednost $\mu(t)$ (usrednjavanjem po ansamblu) i odgovarajuća varijansa $\sigma^2(t)$. Umjesto ovih statistika, posmatrajmo srednju vrijednost ishoda $X(t)$, računatu uzimanjem ishoda dobijenih u susjednim trenucima:

$$\mu_X(t)=\frac{1}{T}\int_{t-T/2}^{t+T/2}X(p)dp.$$

Varijansa predmetnog posmatranja računata po vremenu (u okolini trenutka t):

$$\sigma_X^2(t) = \frac{1}{T}\int_{t-T/2}^{t+T/2} [X(p) - \mu_X(t)]^2\, dp.$$

Procesi kod kojih su srednja vrijednost i varijansa računati po ansamblu jednaki ovim statistikama, računatim usrednjavanjem po vremenu:

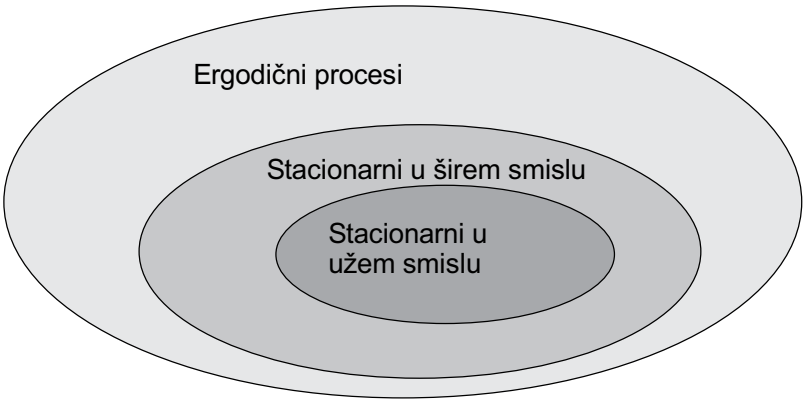
$$\mu_X(t) = \mu(t) \qquad \sigma_X^2(t) = \sigma^2(t),$$

nazivaju se **ergodičnim**. Dakle, kod ergodičnih procesa dozvoljene su promjene ovih statistika, ali su statistike nezavisne od načina na koji je računanje obavljeno (usrednjavanjem po ansamblu ili po vremenu). Svaki proces koji je stacionaran u širem smislu, ujedno je ergodičan, a obrnuto ne mora da važi. Veze pomenutih tipova procesa prikazane su na Slici I.4. U daljem tekstu smatraćemo da (gotovo) svi razmatrani problemi pripadaju makar ergodičnoj klasi procesa, a neergodične procese (zbog složenosti) nećemo ni razmatrati.

U praksi često razmatramo slučajne procese u frekvencijskom domenu pošto je ponašanje šuma na različitim frekvencijama od velikog značaja u komunikacijama. Zbog toga, kao i zbog drugih analiza u statistici, često se razmatra **autokorelaciona funkcija**. Definišimo je i za kontinualne i za diskretne pojave, putem usrednjavanja po vremenskim trenucima:

$$r_{XX}(t,\tau) = \frac{1}{T}\int_{t-T/2}^{t+T/2} [X(p+\tau) - \mu_X(t)][X(p) - \mu_X(t)]dp$$

$$r_{XX}(n,m) = \frac{1}{N}\sum_{k=n-N/2}^{n+N/2-1} [X(k+m) - \mu_X(n)][X(k) - \mu_X(n)].$$



Slika I.4. Dijagram koji prikazuje grupisanje procesa u odgovarajuće klase

Jasna je sada veza između autokorelacije i varijanse:

$$\sigma_X^2(t) = r_{XX}(t,0)$$

$$\sigma_X^2(n) = r_{XX}(n,0).$$

Kod stacionarnih procesa u širem smislu važi da autokorelaciona funkcija ne zavisi od trenutka posmatranja, već samo od rastojanja (pomjeraja) τ u odnosu na trenutak posmatranja: $r_{XX}(t+\tau,t) = r_{XX}(\tau)$, odnosno $r_{XX}(n+m,n) = r_{XX}(m)$. Može se pokazati da je autokorelaciona funkcija maksimalna u centru $r_{XX}(0) \geq |r_{XX}(\tau)|$. Neki od fundamentalnih rezultata vezanih za stohastičke procese izvedeni su u spektralnom (frekventnom ili Furijeovom – fr. *Fourier*) domenu. Spektralna karakterizacija slučajnih procesa može se obaviti, kao i kod bilo kog drugog signala, putem Furijeove transformacije:

$$\Xi_{X,T}(\omega) = \int_{-T/2}^{T/2} X(t)e^{-j\omega t}dt \qquad \Xi_{X,N}(\omega) = \sum_{n=-N/2}^{N/2} X(n)e^{-j\omega n}.$$

Naravno, jedna realizacija slučajnog procesa ili korišćenje kratkog intervala posmatranja ne mogu nam dati realnu sliku kod slučajnih procesa spektralnih karakteristika. Stoga se često koristi **spektralna gustina snage** za Furijeovu transformaciju računatu preko izuzetno dugog vremenskog intervala:

$$S_{XX}(\omega) = \lim_{T\to\infty} \frac{|\Xi_{X,T}(\omega)|^2}{T},$$

gdje $\lim_{T\to\infty}$ predstavlja limes (granična vrijednost) izraza argumenta kada dužina intervala teži beskonačno ($T \to \infty$), odnosno u praksi za veoma dugačak interval opservacije. **Snaga slučajnog procesa** se definiše kao:

$$P_{XX} = \lim_{T\to\infty} \frac{1}{T}\frac{1}{2\pi}\left[\int_{-\infty}^{\infty} |\Xi_{X,T}(\omega)|^2 \, d\omega\right] = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{XX}(\omega)d\omega = \frac{1}{\pi}\int_{0}^{\infty} S_{XX}(\omega)d\omega.$$

Posljednja jednakost slijedi iz činjenice da je spektralna gustina snage parna funkcija $S_{XX}(-\omega) = S_{XX}(\omega)$. Spektralna gustina snage može se dovesti u vezu s autokorelacionom funkcijom:

$$S_{XX}(\omega) = \lim_{T\to\infty} \frac{|\Xi_{X,T}(\omega)|^2}{T} = \lim_{T\to\infty} \frac{1}{T}\Xi_{X,T}(\omega)\Xi_{X,T}^*(\omega) =$$

$$= \lim_{T\to\infty} \frac{1}{T}\int_{-T/2}^{T/2}\int_{-T/2}^{T/2} X(t')X^*(t'')e^{-j\omega t'}e^{j\omega t''}dt'dt'' =$$

$$= \lim_{T\to\infty} \frac{1}{T}\int_{-T/2-t}^{T/2-t}\int_{-T/2}^{T/2} X(t+\tau)X^*(t)e^{-j\omega(t+\tau)}e^{j\omega t}dtd\tau = \lim_{T\to\infty} \frac{1}{T}\int_{-T/2-t}^{T/2-t}\int_{-T/2}^{T/2} X(t+\tau)X^*(\tau)e^{-j\omega t}dtd\tau.$$

Posljednji izraz slijedi na osnovu smjene koordinata. Uzimajući izraz za autokorelaciju, slijedi:

$$S_{XX}(\omega)=\lim_{T\to\infty}\frac{1}{T}\int_{-T/2-t}^{T/2-t}\int_{-T/2}^{T/2}R_{XX}(t+\tau,t)e^{-j\omega\tau}dtd\tau=\int_{-\infty}^{\infty}\lim_{T\to\infty}\frac{1}{T}\left[\int_{-T/2}^{T/2}R_{XX}(t+\tau,t)dt\right]e^{-j\omega\tau}d\tau.$$

Granice prvog (spoljnjeg) integrala su postavljene na $(-\infty,\infty)$ jer u unutrašnjem integralu imamo T koje teži ∞. Izraz:

$$\lim_{T\to\infty}\frac{1}{T}\left[\int_{-T/2}^{T/2}R_{XX}(t+\tau,t)dt\right]$$

predstavlja srednju vrijednost po vremenu. Pod pretpostavkom da je autokorelacija nezavisna od vremenskog trenutka u kome je posmatranje obavljeno, možemo zapisati:

$$R_{XX}(\tau)=\lim_{T\to\infty}\frac{1}{T}\left[\int_{-T/2}^{T/2}R_{XX}(t+\tau,t)dt\right].$$

Stoga dobijamo da je spektralna gustina snage slučajnog procesa:

$$S_{xx}(\omega)=\int_{-\infty}^{\infty}R_{XX}(\tau)e^{-j\omega\tau}d\tau.$$

Dobijena relacija predstavlja poznatu **Viner–Kinčinovu** (engl. *Wiener–Khinchin*) teoremu. Vrijedi napomenuti da je, bez dokaza, ovaj stav uveo Albert Ajnštajn (engl. *Albert Einstein*). Ova teorema je značajna jer uspostavlja vezu između autokorelacione funkcije i spektralne gustine signala, odnosno pokazuje da je spektralna gustina snage slučajnog procesa Furijeova transformacija autokorelacione funkcije, to jest ove dvije veličine predstavljaju Furijeov transformacioni par. Kao i u drugim oblastima tehnike, Furijeova transformacija je značajna jer se često neke pojave lakše analiziraju ili mjere u Furijeovom domenu nego direktno u vremenskom.

I.5 Grupa, polje i vektorski prostor

Već smo rekli u uvodu ovoga poglavlja da je osnovni alat teorije informacija probabilistički, dok je osnovni alat teorije kodova linearna algebra. U okviru algebre definisane su različite **algebarske strukture**. Sa mnogima od njih smo se prirodno upoznali već od prvih razreda osnovne škole, ali se algebarske strukture mogu definisati na mnogo raznovrsnijim skupovima nego što su to brojevi. Najbitnija algebarska struktura je **polje**, dok se mnoge druge mogu izvesti iz polja i to redukujući broj osobina u odnosu na samo polje.

Polje je algebarska struktura koja se definiše nad elementima nekoga skupa – alfabeta koji ćemo označiti sa X. Alfabet može imati beskonačan ili konačan broj elemenata. Polje podrazumijeva da su nad elementima skupa definisane dvije operacije, koje ćemo označiti sa $+$ (uslovno se može nazvati sabiranjem) i $\cdot$ (uslovno se može nazvati množenjem), koje zadovoljavaju sljedećih devet osobina:

Osobina 1 (asocijativnost sabiranja):	za svako $x, y, z \in X$	$x+(y+z)=(x+y)+z$
Osobina 2 (asocijativnost množenja):	za svako $x, y, z \in X$	$x \cdot (y \cdot z)=(x \cdot y) \cdot z$
Osobina 3 (komutativnost sabiranja):	za svako $x, y \in X$	$x+y=y+x$
Osobina 4 (komutativnost množenja):	za svako $x, y \in X$	$x \cdot y=y \cdot x$
Osobina 5 (nulti član):	postoji $0 \in X$	$x+0=0+x=x$
Osobina 6 (jedinični član):	postoji $1 \in X$	$x \cdot 1=1 \cdot x=x$
Osobina 7 (aditivni inverzni element):	za svako $x \in X$ postoji $-x \in X$, tako da:	$x+(-x)=(-x)+x=0$
Osobina 8 (multiplikativni inverz):	za svako nenulto $x \in X$ koje $x \neq 0$ postoji	$x^{-1} \in X, x^{-1} \cdot x=x \cdot x^{-1}=1$
Osobina 9 (distributivnost)	za svako $x, y, z \in X$	$x \cdot (y+z)=x \cdot y+x \cdot z.$

Na primjer, skupovi realnih **R** i kompleksnih brojeva **C** sa standardnim aritmetičkim operacijama sabiranja i množenja su polja. Međutim, polje nije skup cijelih brojeva **N** s operacijama sabiranja i množenja zato što ne postoji adekvatan inverzni element za operaciju množenja (1/2 nije element skupa cijelih brojeva i sa njim se ne može pomnožiti 2 da bi dalo jedinični element).

U primjenama koje se tiču teorije kodova, skupovi od interesa su konačni. Dobar primjer pogodnog skupa koji s odgovarajućim operacijama predstavlja polje je skup cijelih brojeva sa konačnim (prostim) brojem elemenata $\{0, 1, \ldots, p-1\}$, gdje je p **prost broj**. Prost broj je onaj koji je djeljiv bez ostatka samo sa samim sobom i jedinicom. Operacije sabiranja i oduzimanja koje čine sa datim skupom polje definisane su kao:

$$a+b=(a+b)\,\%\,p=\mathrm{rem}(a+b, p)=\mathrm{mod}(a+b, p)$$

$$a\,b=(ab)\,\%\,p=\mathrm{rem}(ab, p)=\mathrm{mod}(ab, p).$$

Ponekad se uvodi posebna oznaka jednakosti $a \equiv b (\mathrm{mod}\ p)$, koja znači da brojevi a i b imaju isti ostatak pri dijeljenju sa p, što se naziva **kongruencijom** (brojevi su kongruentni). Bez namjere da uvedemo konfuziju u označavanju, operacije $+$ i $\cdot$ na lijevoj strani su one koje su asocirane polju i skupu X, dok su ostali opera-

tori standardni algebarski operatori sabiranja i množenja. Oznake %, rem i mod predstavljaju ostatak pri dijeljenju (% u programskim jezicima C, C ++ i srodnim; rem u MATLAB-u; mod u matematičkoj literaturi). Relativno je lako kod ovog sistema uočiti postojanje negativnog broja, odnosno broja koji sabran s elementom skupa daje 0 (osobina 7):

$$x + (-x) = 0 \qquad \text{za } -x = p - x.$$

Na primjer za $p = 7$ i $x = 3$ slijedi $-x = 7 - 3 = 4$. Nešto je teže odrediti multiplikativni inverz. Međutim, za relativno malo p, multiplikativni inverz može se odrediti metodom probe i pokušaja. Tako za posmatrani primjer $p = 7$ slijedi:

$1^{-1} = 1$ jer je $1 \cdot 1 = 1$

$2^{-1} = 4$ jer je $2 \cdot 4 = \text{rem}(2 \cdot 4,7) = \text{rem}(8,7) = 1$

$3^{-1} = 5$ jer je $3 \cdot 5 = \text{rem}(3 \cdot 5,7) = \text{rem}(15,7) = 1$

$6^{-1} = 6$ jer je $6 \cdot 6 = \text{rem}(6 \cdot 6,7) = \text{rem}(36,7) = 1$.

Sistematski način određivanja multiplikativnog inverza je putem male Fermaove (fr. *Fermat*) teoreme. Jasno je, premda nije prikazano, da je $4^{-1} = 2$ i $5^{-1} = 3$ jer mora da važi $(x^{-1})^{-1} = x$.

Polje je najprirodniji tip algebarskog koncepta. Međutim, ovaj koncept nadograđen je nad prostijim konceptom **grupe**. Grupa je definisana nad skupom i jednom operacijom. Zadovoljava samo tri osobine: asocijativnost (osobina 1 ili 2 u zavisnosti od operacije nad kojom je grupa definisana), postojanje neutralnog elementa kojim se ne mijenja operand na koji je operacija primijenjena (nula za operaciju kao što je sabiranje i jedinica za operaciju kao što je množenje – osobine 5 i 6 kod polja) i operaciju inverzije (koja je jedino primjenljiva kod sabiranja – osobina 7, jer multiplikativni inverz nije primjenljiv kod nultog elementa u polju – osobina 8). Dakle, operacija sabiranja, iz polja sa pripadajućim skupom, čini grupu, dok se to ne može reći za operaciju množenja nad skupom X, već samo nad skupom iz koga je eliminisan nulti element $X\backslash\{0\}$ (oznaka \ predstavlja razliku skupova, odnosno elemente koji se nalaze u prvom, ali ne i u drugom skupu operandu).

Abelova ili **komutativna grupa** je ona kod koje, pored osobina koje važe kod grupe, važi i osobina komutativnosti (za sabiranje osobina 3, a množenje osobina 4). **Ciklična grupa** je ona kod koje se svaki element grupe može generisati uzastopnom primjenom operacije iz grupe nad jednim njenim elementom (osim eventualno nultog). Postoji **aditivna ciklična** (definisana nad operacijom sabiranja) i **multiplikativna ciklična grupa** (definisana na operaciji množenja). Za nas su od većeg značaja multiplikativne ciklične grupe kod kojih se svaki element skupa $X\backslash\{0\}$ može dobiti uzastopnom primjenom operatora množenja nad nekim elementom

skupa. Taj element skupa naziva se **generatorom grupe**. U našem slučaju možemo govoriti i o **generatoru polja** pošto ćemo zapravo sve vrijeme posmatrati polja definisana nad operacijama sabiranja i množenja. Posmatrajmo sada skup $X=\{0, 1, 2, 3, 4, 5, 6\}$ sa $p=7$ i operacijom množenja, definisanom putem ostatka pri dijeljenju sa prostim brojem. Za ovaj skup generator je $a=3$. Izvršimo provjeru:

$$a^1=3, \quad a^2=a\cdot a=\text{rem}(3\cdot 3,7)=2, \quad =a^3=a\cdot a\cdot a=\text{rem}(27,7)=6,$$

(uočimo da se a^3 može zapisati i kao $a^2\cdot a=2\cdot 3$)

$$a^4=(a^2)^2=2^2=4,\ a^5=a^4\cdot a=\text{rem}(4\cdot 3,7)=\text{rem}(12,7)=5,$$

$$a^6=\ a^5\cdot a=\text{rem}(5\cdot 3,7)=\text{rem}(15,7)=1=a^0$$

$$a^7=\ a^6\cdot a=1\cdot a=1=3.$$

Dakle, uočili smo da je $a^{p-1}=a$, odakle i potiče cikličnost ove grupe, jer za svaki stepen koji je veći od p dobijemo ponovo članove iz polja, odnosno važi da je:

$$a^r=a^{r\,\%\,(p-1)}.$$

Iz činjenice da se stepeni (elementi skupa) ponavljaju ciklično sa periodom $(p-1)$, jasno slijedi i razlog zbog kojeg se predmetna grupa i naziva cikličnom. Uočimo još jednu činjenicu, da imamo:

$$a^5\cdot a=1.$$

Dakle, ovo ukazuje da je $a^5=a^{-1}$, odnosno da je $a^{p-2}=a^{-1}$ (što je suštinski iskaz male Fermaove teoreme). Ova će osobina biti korišćena u Poglavlju VI, kod kodiranja i dekodiranja Hemingovih kodova.

Posebno pojednostavljen, a veoma bitan tip polja je onaj koji se konstruiše nad binarnim skupom (odnosno alfabetom) $X=\{0, 1\}$. I ovo je skup sa prostim brojem članova $p=2$, ali se operacije mogu provoditi putem logičkih operacija (Bulove algebre − dobila ime po Džordžu Bulu, engl. *George Boole*). Tako operaciji sabiranja odgovara operacija sabiranja po modulu 2, odnosno ekskluzivno ili operacija $0+0=0\,\%\,2=0\oplus 0=0$, $1+0=1\,\%\,2=0\oplus 1=1$, $0+1=1\,\%\,2=1\oplus 0=1$, $1+1=2\,\%\,2=1\oplus 1=0$, dok je množenje konjukcijom (logičkom operacijom i) $0\cdot 0=0\wedge 0=0\,\%\,2=0$, $0\cdot 1=0\wedge 1=0\,\%\,2=0$, $1\cdot 0=1\wedge 0=0\,\%\,2=0$ i $1\cdot 1=1\wedge 1=1\,\%\,2=1$. Pošto je binarni zapis kodova najčešći, ujedno je ovakva algebra, odnosno korespondentna polja i grupe, i najvažnija, te će biti najviše korišćena u ovoj knjizi. Istina, s vremena na vrijeme ćemo se otisnuti i u svijet nebinarnih kodova, i kod kodiranja izvora i kod kodiranja kanala.

Vektorski prostori su naredni bitan pojam kada govorimo o algebarskim strukturama. Vektorski prostor je suštinski domen u kome se nalaze svi mogući vektori

koji zadovoljavaju neki uslov. Vektori mogu da imaju određeni broj koordinata (dimenzija). Pogodno ih je prikazati putem skupa uređenih *n*-torki:

$$[a_1, a_2, \ldots, a_n] \qquad [a_0, a_2, \ldots, a_{n-1}]$$
$$\{a_1, a_2, \ldots, a_n\} \qquad \{a_0, a_2, \ldots, a_{n-1}\},$$

gdje svaki element a_i, $i = 1, 2, \ldots, n$ (indeksiranje može početi i od nule a_i, $i = 0, 2, \ldots, n-1$) pripada nekom skupu ili gdje se nad elementima ovoga prostora mogu uvesti određena ograničenja (npr. da se nalaze u nekoj *n*-dimenzionoj sferi itd.). Članovi skupa koji predstavlja vektorski prostor mogu se definisati nad nekim poljem. Na primjer, mogu biti polje koje smo uveli sa prostim brojem elemenata i operacijama sabiranja i množenja po modulu. Ovakvo polje se često označava F_p, gdje je *p* prost broj (*F* potiče od engl. *field* – polje). Vektorski prostor, uveden nad ovakvim poljem, označava se kao $V_n(F_p)$ ili $V(p^n)$ (*V* se često podrazumijeva od riječi zapremina – *volume*, dok je indeks *n* dimenzija prostora). Pored vektorskog označavanja elemenata polja, koristi se i notacija u obliku polinoma:

$$\mathbf{a}(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}.$$

Sada je i jasnije zašto je pogodniji zapis s indeksiranjem koje počinje s nultim indeksom nego s indeksom 1.

Vektorski prostori se, po pravilu, dizajniraju da zadovoljavaju osobine algebarskih struktura. Dakle, ako imamo skup svih vektora u vektorskom prostoru $V_n(F_p)$ ili neki podskup ovog vektorskog prostora koji zadovoljava neku pogodnu osobinu, kako definisati operacije sabiranja i množenja nad ovim poljem/vektorskim prostorom? Operacija sabiranja je očigledna. Ako imamo dva vektora u navedenom prostoru, ona se može obaviti na prirodan način − sabiranjem koeficijenata u skladu s algebrom iz konačnog polja F_p, nad kojim je definisan vektorski prostor $V_n(F_p)$:

$$\mathbf{a}(x) + \mathbf{b}(x) = (a_0 + b_0) + (a_1 + b_1)x + \cdots + (a_{n-1} + b_{n-1})x^{n-1} =$$
$$= \mathrm{rem}(a_0 + b_0, p) + \mathrm{rem}(a_1 + b_1, p)x + \cdots + \mathrm{rem}(a_{n-1} + b_{n-1}, p)x^{n-1}.$$

Značajno složenija operacija je množenje. I ona se može definisati „skalarnim“ proizvodom po pojedinim koeficijentima vektora (svaki s odgovarajućim, uz član istoga stepena). Međutim, to nije u skladu sa standardnim množenjem polinoma, a što je mnogo bitnije − ovakva operacija nije od koristi ni u skladu s našim aktivnostima vezanim za kodiranje (kanala). Na primjer, kod standardnog množenja polinoma poznato je da je rezultujući polinom većeg stepena nego što su polinomi činioci (stepen rezultujućeg polinoma jednak je zbiru stepena polinoma činioca), tako da bi u tom slučaju bile narušene osnovne osobine polja (preciznije, grupe) da rezultat operacije mora da ostane u istom skupu kao što je skup operanada.

Posmatrajmo proizvod dva polinoma $n-1$ reda:

$$\mathbf{a}(x)\mathbf{b}(x)=\sum_{k=0}^{n-1}\sum_{r=0}^{n-1}a_k b_r x^{k+r},$$

gdje je suma proizvoda $\sum_{k=0}^{n-1}\sum_{r=0}^{n-1}a_k b_r$ računata po modulu p. Predmetni polinom je stepena $2n-2$ i ne nalazi se u polju $V_n(F_p)$. Najčešći način da se predmetni rezultat zadrži u vektorskom prostoru $V_n(F_p)$ jeste da se ovaj proizvod takođe posmatra po modulu (ostatku pri dijeljenju) sa nekim polinomom n reda. Na primjer, ako je $n-1=2$, proizvod dva polinoma će biti četvrtog reda, a ostatak pri dijeljenu s polinomom trećeg reda ponovo će biti drugog reda. Najčešće korišćeni polinomi po kojima se računa moduo (ostatak pri dijeljenju) imaju oblik x^M-1, s tim da se u našoj aritmetici po konačnom polju vrijednost izraza ne mijenja ako se izrazu doda p, tako da se ovaj polinom može zapisati i kao: x^M+p-1.

Primjer I.3. Za vektorski prostor $V_3(F_2)$, konstruisan nad binarnim alfabetom $\{0,1\}$, sa vektorima dužine $n=3$ odrediti proizvod polinoma x^2+x+1 i x^2 po modulu x^3-1.

Rješenje: Polinom x^3-1 se u ovoj algebri može napisati i kao x^3+1. Proizvod činilaca je:

$$(x^2+x+1)x^2=x^4+x^3+x^2.$$

Podijelimo dobijeni proizvod sa polinomom x^3+1:

$$\begin{array}{l}(x^4+x^3+x^2):(x^3+1)=x+1\\ \underline{x^4+\quad\quad x}\\ \quad x^3+x^2+x\\ \quad\underline{x^3+\quad\quad 1}\\ \quad\quad x^2+x+1.\end{array}$$

Prilikom provođenja operacija uočljivo je da je u ovoj algebri s koeficijentima iz binarnog alfabeta operacija sabiranja identična s operacijom oduzimanja, tako da je oduzimanje člana x isto kao i njegovo sabiranje. Kod većih konačnih polja (za veće p) ovo ne mora da važi. Konačno, rezultat ove operacije je ostatak pri dijeljenju x^2+x+1, koji je u ovom slučaju jednak jednom od polaznih polinoma činilaca. □

Najpoznatiji rezultat u ovoj oblasti su **Galoaova polja**. Polja su dobila ime po poznatom francuskom matematičaru Evaristu Galoau (fr. *Evarist Galois*), često imenovanom princu matematike, kojem je život okončan u dvoboju, u reveolucionarnim godinama prve polovine XIX vijeka, u dvadesetoj godini života! Život ovog naučnika i revolucionara ponekad se naziva „bajkom za naučnike“. Da bismo pokušali makar malo da uđemo u problematiku ovih veoma bitnih algebarskih struktura, pretpostavimo da imamo polje sa $p=3$ elementa $\{0, 1, 2\}$ i odgovara-

jućim operacijama sabiranja i množenja, definisanim preko operacije po modulu $p=3$. Uvedimo i operaciju oduzimanja:

$$a-b=a+(-b).$$

Inverzni element za operaciju sabiranja može se definisati kao $-b=p-b$, pa dobijamo:

$$a-b=\mathrm{rem}(a+p-b,p).$$

Formirajmo sada polinom koji ima tri nule koje su elementi posmatranog skupa:

$$f(x)=x(x-1)(x-2)=x^3-3x^2+2x=\ x^3+(3-3)x^2+(2-3)x=x^3-x.$$

Dakle, polinom $f(x)=x^p-x$ ima nule koje su svi elementi iz skupa. Slično, nule polinoma:

$$g(x)=x^{p-1}-1$$

su svi nenulti elementi skupa. Ovo važi za sva polja s prostim brojem elemenata koja su uvedena na predmetni način i za operacije sabiranja i množenja koje su definisane po modulu prostog broja. Provjerite, na primjer, za $p=7$! Ovaj stav se može i eksplicitno dokazati putem tzv. male Fermaove teoreme, ali vam dokaz ostavljamo za vježbu i proučavanje odgovarajuće literature. Još jedan interesantan stav koji važi u ovom polju je:

$$(x+y)^p=x^p+y^p.$$

U literaturi se naziva i „brucoškim snom“ (engl. *freshman's dream*). U standardnoj algebri, nad skupovima cijelih i realnih brojeva ovo ne važi, a u konačnom polju je zadovoljeno, što se lako dokazuje razvojem binomnog obrasca:

$$(x+y)^p=\sum_{k=0}^{p}\binom{p}{k}x^k y^{p-k}=x^p+y^p+\sum_{k=1}^{p-1}\binom{p}{k}x^k y^{p-k}=$$

$$=x^p+y^p+\sum_{k=1}^{p-1}\frac{p!}{(p-k)!k!}x^k y^{p-k}.$$

Član u sumi $p!/[(p-k)!k!]$ ima brojilac koji je djeljiv sa p, dok imenilac zasigurno nije djeljiv sa p pošto su u njemu svi članovi manji od p i p je prost broj. Pošto u usvojenoj algebri važi $a=a\,\%\,p$, to dalje slijedi da je $p=p\,\%\,p=\mathrm{rem}(p,p)=0$, tako da

$$\sum_{k=1}^{p-1}\frac{p!}{(p-k)!k!}x^k y^{p-k}=0,$$

odnosno $(x+y)^p=x^p+y^p$.

Kod Galoaovih polja broj elemenata koji se mogu konstruisati jednak je p^n, gdje je p prost broj, odnosno broj članova ovog polja nije prost. Stoga se često Galoaova polja označavaju kao $GF(p^n)$ (G od prezimena, a F od polje − engl. *field*). Predmetna polja se mogu konstruisati na više načina. Ovdje ćemo to prikazati na način koji će nam kasnije odgovarati prilikom konstrukcije kodova. Bitan element u konstrukciji ovih polja igraju **nesvodivi polinomi**, koji se ponekad pogrešno nazivaju i **prostim**. Naime, nesvodivi polinomi posjeduju bitnu osobinu da nisu bez ostatka djeljivi ni sa jednim drugim polinomom manjega reda, osim sa jedan i sa samim sobom. Da bi polinom bio nazvan prostim, on mora u ovoj terminologiji da zadovoljava i dodatnu osobinu.

Definicija I.1. Polinom se naziva prostim u Galoaovom konačnom polju GF(p^n) ako je nesvodiv i ako dijeli bez ostatka polinom:

$$x^{p^n-1}-1,$$

dok ne dijeli bez ostatka nijedan polinom x^q-1 za $q<p^n-1$. □

Nesvodivi polinom u polju $GF(2^4)$ x^4+x+1 je prost jer ne dijeli bez ostatka nijedan polinom x^q-1 za $q<p^n-1=15$, dok dijeli polinom $x^{15}-1$ bez ostatka. Međutim, nesvodivi polinom $x^4+x^3+x^2+x+1$ nije prost jer bez ostatka dijeli polinom x^q-1 manjeg reda od $q=15$, odnosno dijeli polinom x^5-1 (odnosno x^5+1) bez ostatka:

$$x^5+1:x^4+x^3+x^2+x+1=x+1$$

jer je $x^5+1=(x^4+x^3+x^2+x+1)(x+1)=x^5+x^4+x^3+x^2+x+x^4+x^3+x^2+x+1=x^5+1.$

Prosti polinomi su od izuzetnog značaja u procesu konstrukcije kodova za kodiranje kanala, što će biti jasnije u Poglavlju VI.

I.6 Euklidski algoritam

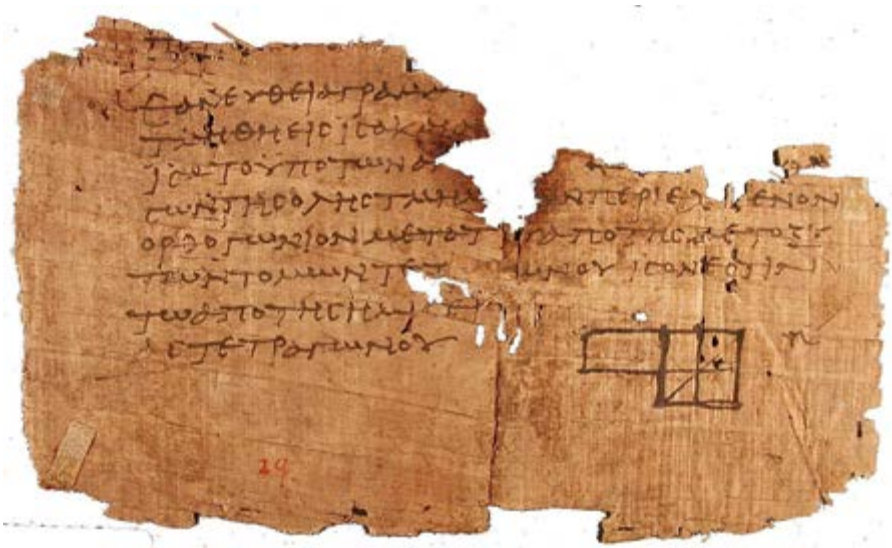
I.6.1 Istorija i osnovni oblik algoritma

Euklid iz Aleksandrije je helenski mislilac i matematičar iz III vijeka prije nove ere. Smatra se tvorcem geometrije. Znamo da je i prije njega postojao utemeljen metodološki pristup u matematici, ali smo kroz njegovo djelo dobili zaokružen i veoma savremen naučni metod u kojem se polazi od očiglednih činjenica (aksioma), nad kojima se zatim nadograđuju (i dokazuju) drugi stavovi, koje nazivamo teoremama, lemama, tvrdnjama i posljedicama (u zavisnosti od složenosti i predmeta stava). Stoga se u nekoj mjeri Euklid može smatrati i tvorcem naučnog metoda, odnosno tvorcem sistematičnog bavljenja naukom. Euklidovo djelo je uglavnom stiglo do savremenog čovjeka u prepisima, a najobimniji su oni arapski. Međutim, najstariji sačuvani fragmenti potiču iz I vijeka nove ere. Poznati su

papirusi iz Oksirinhusa. Ovo je grad na obalama rijeke Nil u Donjem Egiptu, gdje je postojala, relativno nedavno otkrivena, deponija koja sadrži oštećene papiruse koji su se na tom mjestu odlagali nekoliko vjekova. Pisani su različitim jezicima (u to doba grčki je bio *lingua franca* antičkog svijeta), a predstavljaju gotovo sve životne aktivnosti (poreze, finansije, medicinu, religiju itd.). Posebno je značajno da su u ovom nalazištu nađeni djelovi brojnih starozavjetnih rukopisa, kao i neki od najstarijih prepisa novozavjetnih knjiga. Među ostalima, nađeni su i naučni tekstovi, među kojima je jedan iz najpoznatnijeg Euklidovog djela, iz knjige „Elementi“, koji je prikazan na Slici I.5.

Premda je Euklid tvorac geometrije, brojne tvrdnje koje je iznio tipično su algebarske (dokazivao ih je geometrijski, putem duži i sličnih geometrijskih koncepata). Za nas je značajna Tvrdnja II iz Knjige VII, koja za dva cijela broja, a i b, određuje najveći zajednički djelilac (nzd) (engl. *greatest common divisior* – gcd), zapisano u smislu moderne algebre:

$$\begin{aligned} \text{nzd}(a,b) &= \text{gcd}(a,b) \\ \text{gcd}(a,b) &= \text{gcd}(a-b,b) \quad \text{za } a \geq b \\ \text{gcd}(a,b) &= \text{gcd}(b,a) \qquad \text{za } a < \text{b} \\ \text{gcd}(a,0) &= a. \end{aligned}$$



Slika I.5. Fragment iz papirusa broj 29, nađen 1897. godine u Oksirinhusu, koji predstavlja dio druge knjige „Elemenata“, tvrdnju broj 5. Papirus je datiran u I vijek nove ere (oko 400 godina nakon Euklida)

Primjer I.4. Odrediti najveći zajednički djelilac brojeva $a = 9$ i $b = 57$.

Rješenje: $\gcd(9,57) = \gcd(57,9) = \gcd(57-9,9) = \gcd(48,9) = \gcd(39,9) =$
$= \gcd(30,9) = \gcd(21,9) = \gcd(12,9) = \gcd(3,9) = \gcd(9,3) =$
$= \gcd(9-3,3) = \gcd(6,3) = \gcd(3,3) = \gcd(0,3) = \gcd(3,0) = 3.$ □

Kao što je vjerovatno poznato Euklidski algoritam je popularan primjer rekurzivnog izvršavanja funkcija u programiranju. Oduzimamo manji argument od većega, dok se ne svede na veličinu koja je manja od argumenta koji je bio na početku manji. Zatim argumenti zamijene mjesta i procedura se ponavlja oduzimanjem manjeg od većeg argumenta. Kada se dobije da je jedan od argumenata jednak nuli, onda je najveći zajednički djelilac jednak preostalom argumentu.

U Primjeru I.4 vidimo uzastopno umanjenje većeg argumenta, koje se može efikasnije provesti putem operacije računanja ostatka pri dijeljenju. Procedura za računanje najvećeg zajedničkog djelioca može se zapisati kao:

$$\gcd(a,b) = \begin{cases} \gcd(b,a) & a < b \\ \gcd(b, a\%b) & a \geq b \\ a & b = 0. \end{cases}$$

Primjer I.4 se sada može sračunati nešto brže, kao:

$$\gcd(9,57) = \gcd(57,9) = \gcd(9,57\,\%\,9) = \gcd(9,3) = \gcd(3,0) = 3,$$

gdje smo koristili programerski operator $a\,\%\,b$ za ostatak pri dijeljenju.

I.6.2 Euklidski algoritam za cijele brojeve

Pored osnovne namjene Euklidskog algoritma za određivanje najvećeg zajedničkog djelioca, s vremenom su uočene brojne posljedice ovoga postupka, koje po značaju daleko prevazilaze prvobitnu namjenu. Neke od posljedica Euklidskog algoritma su od izuzetnog značaja u teoriji kodova. Za sva cijela boja a i b važi:

$$a = q \cdot b + r,$$

gdje je q cjelobrojni količnik, dok je r ostatak pri dijeljenju. Najveći zajednički djelilac a i b može se zapisati kao linearna kombinacija a i b:

$$\gcd(a,b) = u\,a + v\,b.$$

Na primjer, za $a = 52$ i $b = 32$ težine su $u = -3$ i $v = 5$ i daju najveći zajednički djelilac:

$$\gcd(52,32) = -3 \cdot 52 + 5 \cdot 32 = 4,$$

dok se za $a = 910$ i $b = 143$ najveći zajednički djelilac dobija za težinske koeficijente $u = 3$ i $v = -19$:

$$\gcd(910,143) = 3 \cdot 910 - 19 \cdot 143 = 13.$$

Postavlja se pitanje: kako se dobijaju koeficijenti u i v? To se obavlja rekurzivnom procedurom koja je slična samom Euklidskom algoritmu. Inicijalizuju se vrijednosti na $r_{-1} = a$ i $r_0 = b$. U svakoj narednoj iteraciji formira se novo r_k, u_k i v_k tako da važi linearna kombinacija:

$$r_k = u_k\, a + v_k\, b.$$

Prvo se određuje vrijednost r_k, koja je ostatak pri dijeljenju r_{k-1} i r_k, tako da važi (niz r je niz ostataka u dijeljenju i odatle oznaka r od engl. *remainder* – ostatak):

$$r_{k-1} = q_{k+1}\, r_k + r_{k+1},$$

gdje je q_{k+1} količnik r_{k-1} i r_k, dok je r_{k+1} ostatak pri dijeljenju. Pretpostavimo da u koracima k i $k+1$ važi (ovo će biti uslovi koji će na početku biti obezbijeđeni na osnovu inicijacije algoritma):

$$r_k = u_k\, a + v_k\, b$$
$$r_{k-1} = u_{k-1}\, a + v_{k-1}\, b.$$

Kada uvrstimo ove izraze u relaciju za ostatak, u koraku $k+1$ dobijamo:

$$r_{k+1} = r_{k-1} - q_{k+1} r_k = u_{k-1} b + v_{k-1} b - q_{k+1}(u_k a + v_k b) = (u_{k-1} - q_{k+1} u_k)\, a + (v_{k-1} - q_{k+1} v_k) b.$$

Zapišimo sada:

$$r_{k+1} = u_{k+1}\, a + v_{k+1}\, b,$$

odakle slijede rekurzivne relacije:

$$u_{k+1} = u_{k-1} - q_{k+1} u_k \qquad\qquad v_{k+1} = v_{k-1} - q_{k+1} v_k.$$

Inicijalizacija algoritma obavlja se tako što se za prve dvije iteracije postavi $u_{-1} = 1$, $u_0 = 0$, $v_{-1} = 0$ i $v_0 = 1$.

Posmatrajmo primjer sa $a = 910$ i $b = 143$. Na osnovu prethodnog algoritma, možemo formirati Tabelu I.4. Na primjer:

$$q_1 = r_{-1}/r_0 = 6 \qquad r_1 = r_{-1} - q_1 r_0 = 910 - 143 \cdot 6 = 52$$
$$u_1 = u_{-1} - q_1 u_0 = 1 - 6 \cdot 0 = 1 \qquad v_1 = v_{-1} - q_1 v_0 = 0 - 6 \cdot 1 = -6$$
$$q_2 = r_0/r_1 = 2 \qquad r_2 = r_0 - q_2 r_1 = 143 - 52 \cdot 2 = 39$$
$$u_2 = u_0 - q_2 u_1 = 0 - 2 \cdot 1 = -2 \qquad v_2 = v_0 - q_2 v_1 = 1 - 2 \cdot (-6) = 13.$$

K	*Q*	*R*	*U*	*V*
-1		910	1	0
0		143	0	1
1	6	52	1	-6
2	2	39	-2	13
3	1	13	3	-19
4	3	0	-11	70

Tabela I.4. Koraci u Euklidskom algoritmu za dva cijela broja *a*=910 i *b*=143

Ostatak provjerite sami. Kada u nekoj vrsti dobijemo ostatak $r_k = 0$, zaključujemo da je r_{k-1} najveći zajednički djelilac dva broja nad kojima je Euklidski algoritam proveden. Između pojedinih međukoraka u Euklidskom algoritmu mogu se uspostaviti veoma interesantne relacije. Ovdje dajemo tri, za koje smatramo da su najvažnije:

$$r_{k-1}u_k - r_k u_{k-1} = \pm b \qquad r_{k-1}v_k - r_k v_{k-1} = \pm a$$
$$u_{k-1}v_k - u_k v_{k-1} = \pm 1.$$

Ovdje dokazujemo samo prvu od njih, putem matematičke indukcije, dok se dokazi ostalih nalaze u urađenim primjerima na kraju poglavlja. Pođimo od $k = 0$, tada bi trebalo da važi:

$$r_{-1}u_0 - r_0 u_{-1} = -b$$

uzimajući početne uslove u Euklidskom algoritmu. Kako je po indukciji uobičajeno, pokazaćemo da ako važi predmetna relacija za *k*, onda mora da važi i za $k+1$. Dakle, pod pretpostavkom da važi:

$$r_{k-1}u_k - r_k u_{k-1} = \pm b,$$

treba da dokažemo da važi:

$$r_k u_{k+1} - r_{k+1}u_k = \pm b.$$

Kako je $r_{k-1} = q_{k+1}\, r_k + r_{k+1}$, slijedi da je $r_{k+1} = r_{k-1} - q_{k+1} r_k$, odnosno da je $u_{k+1} = u_{k-1} - q_{k+1}u_k$, pa dobijamo:

$$r_k(u_{k-1} - q_{k+1}u_k) - (r_{k-1} - q_{k+1}r_k)u_k = (r_k u_{k-1} - r_{k-1}u_k) - q_{k+1}(r_k u_k - r_k u_k)$$
$$= -(r_{k-1}u_k - r_k u_{k-1}) = \mp b,$$

čime je dokazana predmetna osobina.□

Apostrofirajmo da za red Tabele I.4 važi $r_k = u_k a + v_k b$. Ova osobina važi i u posljednjem redu, gdje se r_k anulira, pa slijedi da je $u_k a + v_k b = 0$, odnosno važi da se v_k/u_k redukuje na $-a/b$.

Važno je još uočiti da apsolutna vrijednost brojeva u koloni ***R*** strogo opada, dok je u kolonama ***U*** i ***V*** strogo rastuća, počevši od vrste $k=1$.

I.6.3 Euklidski algoritam za polinome

Euklidski algoritam se može primijeniti i kod polinoma sa koeficijentima koji su definisani nad bilo kojim pogodnim poljem. Posmatrajmo primjer dva monična polinoma (polinoma s jediničnim koeficijentom uz član najvišeg stepena) koja su definisana koeficijentima u skupu realnih brojeva.

Primjer I.5. Odrediti najveći zajednički djelilac za polinome:

$$a(x)=x^6+2x^5+4x^4+7x^3+8x^2+6x+2$$

$$b(x)=x^3-1.$$

U Tabeli I.5 dat je postupak Euklidskog algoritma analogan onome koji je rađen za cijele brojeve. Najveći zajednički djelilac (sveden na monični polinom dijeljenjem sa 10) je x^2+x+1.□

U teoriji informacija i kodova mnogo su interesantniji polinomi konstruisani nad konačnim poljima. Posmatrajmo vektorski prostor definisan polinomima sa koeficijentima u binarnom polju i operacijama sabiranja i oduzimanja, definisanim preko algebre sa modulom 2.

Primjer I.6. Data su dva polinoma $a(x)=x^7+x^5+x^3+x^2+x+1$ i $b(x)=x^4+x^3+x^2+1$. Koeficijenti ova dva polinoma su binarni brojevi s operacijama i algebrom koje se nad njima provode po modulu 2 (odnosno množenje koeficijenata logičkom operacijom „i“ i sabiranje koeficijenata logičkom operacijom ekskluzivno ili).

Rješenje: Radi kompaktnijeg zapisa, polinome smo, u Tabeli I.6, prikazali putem vektora [1, 0, 1, 0, 1, 1, 1, 1] i [1, 1, 1, 0, 1] koji počinju od koeficijenta najvišeg stepena i idu ka manjim. Vidjećemo kasnije, u Poglavlju VI, da postoje još kompaktniji zapisi. Iz tabele slijedi da je najveći zajednički djelilac ova dva polinoma $x+1$ (polinom ***R*** u $k=3$ vrsta tabele, $r_3(x)$). □

k	***Q***	***R***	***U***	***V***
-1		$x^6+2x^5+4x^4+7x^3+8x^2+6x+2$	1	0
0		x^3-1	0	1
1	x^3+2x^2+4x+8	$10x^2+10x+10$	1	$-x^3-2x^2-4x-8$
2	$(x-1)/10$	0	$-(x-1)/10$	$(x^4+x^3+2x^2+4x-8)/10$

Tabela I.5. Koraci u Euklidskom algoritmu za polinome sa koeficijentima u skupu cijelih brojeva

k	Q	R	U	V
-1		[1, 0, 1, 0, 1, 1, 1, 1]	[1]	[0]
0		[1, 1, 1, 0, 1]	[0]	[1]
1	[1, 1, 1, 0]	[1, 0, 0, 1]	[1]	[1, 1, 1, 0]
2	[1, 1]	[1, 1, 0]	[1, 1]	[1, 0, 0, 1, 1]
3	[1, 1]	[1, 1]	[1, 0, 0]	[1, 1, 1, 0, 1, 1]
4	[1]	[0]	[1, 1, 1]	[1, 0, 1, 0, 0, 0]

Tabela I.6. Euklidski algoritam za polinome sa koeficijentima definisanim u binarnom polju iz Primjera I.6

k	Q	R	U	V
-1		[1, 0, 0, 0, 0, 0, 0, 0, 0]	[1]	[0]
0		[1, 0, 1, 0, 1, 1, 1]	[0]	[1]
1	[1, 0, 1]	[1, 0, 1, 1]	[1]	[1, 0, 1]
2	[1, 0, 0, 1]	[1, 0, 0]	[1, 0, 0, 1]	[1, 0, 1, 1, 0, 0]
3	[1, 0]	[1, 1]	[1, 0, 0, 1, 1]	[1, 0, 1, 1, 1, 0, 1]
4	[1, 1]	[1]	[1, 1, 1, 1, 0, 0]	[1, 1, 0, 0, 1, 0, 1, 1]
5	[1,1]	[0]	[1, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 1, 1, 1]

Tabela I.7. Euklidski algoritam za polinome sa koeficijentima definisanim u binarnom polju (Primjer I.7)

Primjer I.7. Euklidskim algoritmom odrediti najveći zajednički djelilac polinoma $a(x) = x^8$ i $b(x) = x^6 + x^4 + x^2 + x + 1$ sa koeficijentima u binarnom polju.

Rješenje: Očigledno je, i bez provođenja Euklidskog algoritma, da su predmetni polinomi uzajamno prosti jer prvi polinom može biti djeljiv samo sa stepenima x, dok drugi polinom nije djeljiv sa x^k za $k > 0$ jer ima nenulti član uz x^0. Međutim, algoritam ćemo provesti zbog izuzetnog značaja ovog postupka kod navedenih polinoma i primjenljivosti kod kodova, koji će sa više detalja biti izučavani u Poglavlju VI, u Tabeli I.7. □

Uočimo da, kao što kod cijelih brojeva opadaju vrijednosti u koloni $\boldsymbol{R}$, a rastu u kolonama $\boldsymbol{V}$ i $\boldsymbol{U}$, u slučaju polinoma dolazi do smanjivanja reda polinoma u koloni $\boldsymbol{R}$, dok red polinoma u kolonama $\boldsymbol{V}$ i $\boldsymbol{U}$ strogo raste.

Euklidski algoritam je jedan od najznačajnijih postupaka za dekodiranje kodova za kodiranje kanala koji su u stanju da isprave greške koje se dese u kanalu.

Ovim smo jednim dijelom završili s elementarnim matematičkim pregledom pojmova i koncepata neophodnih za praćenje oba krucijalna dijela našeg kursa: teorije informacija (dominantno vezane za probabilističke koncepte) i teorije kodova (dominantno vezane za naprednu algebru). Premda bi se o oba ova alata moglo mnogo više reći ipak ćemo se zaustaviti na ovom nivou, jer nam nije cilj matematička rigoroznost, već samo dovoljnost matematičkih koncepata za razumijevanje svih postupaka i algoritama koji su obuhvaćeni u ovom udžbeniku.

I.7 Zadaci i softverska realizacija

I.7.1 Riješeni zadaci

1.1. Slučajna promjenljiva ξ ima dvije moguće vrijednosti 0 i 1. Vjerovatnoća pojave 0 je jednaka q, dok je vjerovatnoća pojave 1 jednaka p. Odrediti funkciju raspodjele ove funkcije. Odrediti srednju vrijednost i varijansu.

Rješenje: Za $x \geq 1$ događaj $\{\xi \leq x\}$ je siguran, pa je: $P\{\xi \leq x\} = 1$. Za $0 \leq x < 1$ vjerovatnoća događaja $\{\xi \leq x\}$ jednaka je vjerovatnoći $\{\xi = 0\}$: $P\{\xi \leq x\} = P\{\xi = 0\} = q$. Događaj $x < 0$ je nemoguć $\{\xi \leq 0\}$, pa važi: $P\{\xi \leq x\} = 0$. Ovo se ukratko može napisati kao:

$$P_{\xi}(x) = \begin{cases} 1 & x \geq 1 \\ q & 0 \leq x < 1 \\ 0 & x \leq 0. \end{cases}$$

Srednja vrijednost ovog događaja jednaka je $E\{\xi\} = 0 \cdot q + 1 \cdot p = p$, dok je varijansa jednaka: $E\{(\xi - E\{\xi\})^2\} = E\{\xi^2\} - E^2\{\xi\} = p - p^2$. Imajte na umu da je $q + p = 1$.

1.2. U binarnom kanalu se prenose cifre 0 i 1. Ove dvije cifre se pojavljuju sa vjerovatnoćom 1/2. Zbog prisustva smetnji, moguće je da ne bude primljena cifra koja je poslata. Vjerovatnoća da je primljena ispravna cifra 0, ako je poslata 0, jednaka je $P(0|0) = p$, dok je vjerovatnoća da se primi 1, kada je poslato 1, jednaka: $P(1|1) = q$. Odrediti funkciju raspodjele primljenog signala.

Rješenje: Na mjestu prijema, vrijednost nula se pojavljuje u dva slučaja: poslata je nula koja je uspješno prenesena; poslata je jedinica i došlo je do greške. Ovo se može zapisati kao:

$$P\{\xi = 0\} = P(0)P(0\,|\,0) + P(1)P(0\,|\,1) = \frac{1}{2}p + \frac{1}{2}(1-q) = \frac{1}{2}(p - q + 1).$$

Vjerovatnoća događaja da je primljena jedinica je:

$$P\{\xi = 1\} = P(1)P(1\,|\,1) + P(0)P(1\,|\,0) = \frac{1}{2}q + \frac{1}{2}(1-p) = \frac{1}{2}(q - p + 1).$$

Funkcija gustine raspodjele se može zapisati kao:

$$P_{\xi}(x) = \frac{1}{2}(q - p + 1)u(x) + \frac{1}{2}(p - q + 1)u(x-1),$$

gdje je $u(x)$ jedinična diskretna Hevisajdova odskočna funkcija:

$$u(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0. \end{cases}$$

1.3. Slučajni signal ξ, uniformno raspoređen na intervalu $[0,a]$ ulazi u kvantizator koji ima karakteristiku: $y = ns$ za $ns < x \le (n+1)s$, gdje je n cijeli broj, dok je s data konstanta $s=a/5$. Odrediti očekivanu vrijednost slučajne promjenljive na izlazu iz kvantizatora.

Rješenje: Slučajna promjenljiva na izlazu iz kvantizatora je diskretna, sa mogućim vrijednostima: 0, s, $2s$, $3s$ i $4s$. Odgovarajuće vjerovatnoće su:

$$P\{\eta = 0\} = P\{0 < \xi \le s\} = \frac{1}{5}$$

$$P\{\eta=1\}=1/5 \quad P\{\eta=2\}=1/5 \quad P\{\eta=3\}=1/5 \quad P\{\eta=4\}=1/5.$$

Dakle, očekivana vrijednost je:

$$\mathrm{E}\{\eta\} = \sum_{i=1}^{5} P(y_i) y_i = 0\frac{1}{5} + s\frac{1}{5} + 2s\frac{1}{5} + 3s\frac{1}{5} + 4s\frac{1}{5} = 2s.$$

1.4. Diskretna slučajna promjenljiva uzima vrijednosti: x_1, x_2, x_3, x_4 i x_5 sa vjerovatnoćama: 0.1, 0.2, 0.3, 0.3, 0.1. Odrediti funkciju raspodjele, srednju vrijednost i varijansu.

Rješenje: Funkcija raspodjele se može zapisati kao (pod pretpostavkom $x_i < x_{i+1}$):

$$P(\xi) = \begin{cases} 0 & \xi < x_1 \\ 0.1 & x_1 \le \xi < x_2 \\ 0.3 & x_2 \le \xi < x_3 \\ 0.6 & x_3 \le \xi < x_4 \\ 0.9 & x_4 \le \xi < x_5 \\ 1 & x_5 \le \xi. \end{cases}$$

Srednja vrijednost je:

$$\mu_x = 0.1x_1 + 0.2x_2 + 0.3x_3 + 0.3x_4 + 0.1x_5.$$

Varijansa je jednaka:

$$\sigma_x^2 = \mathrm{E}\{(x-\mu_x)^2\} = \mathrm{E}\{x^2\} - \mu_x^2 =$$
$$= 0.1x_2^2 + 0.2x_2^2 + 0.3x_3^2 + 0.3x_4^2 + 0.1x_5^2 - (0.1x_1 + 0.2x_2 + 0.3x_3 + 0.3x_4 + 0.1x_5)^2.$$

1.5. Diskretna slučajna promjenljiva ξ uzima vrijednosti: 2, 4, 6, 8 i 10 sa vjerovatnoćama 0.1, 0.3, 0.4, 0.1, 0.1. Odrediti raspodjelu i očekivanu vrijednost za slučajnu promjenljivu $\eta = \max\{\xi,3\}$. Odrediti varijansu slučajne promjenljive $\rho = \xi + 1$.

Rješenje: Slučajna promjenljiva η ima raspodjelu datu na sljedeći način: vrijednosti 3, 4, 6, 8 i 10 pojavljuju se s vjerovatnoćama 0.1, 0.3, 0.4, 0.1, 0.1. Srednja vrijednost je jednaka:

$$\mu_\eta = 0.1\cdot 3 + 0.3\cdot 4 + 0.4\cdot 6 + 0.1\cdot 8 + 0.1\cdot 10 = 5.7.$$

Varijansa slučajne promjenljive ρ je:

$$\mathrm{E}\{(\rho-\mu_\rho)^2\} = \mathrm{E}\{\rho^2\} - \mu_\rho^2 = \mathrm{E}\{\xi^2 + 2\xi + 1\} - \mu_\rho^2 =$$
$$= \mathrm{E}\{\xi^2\} + 2\mathrm{E}\{\xi\} + 1 - \mu_\rho^2 = \mathrm{E}\{\xi^2\} + 2\mu_\xi + 1 - \mu_\rho^2.$$

Srednja vrijednost slučajne promjenljive ρ je:

$$\mu_\rho = \mathrm{E}\{\rho\} = \mathrm{E}\{\xi + 1\} = \mathrm{E}\{\xi\} + 1 = \mu_\xi + 1,$$

pa odavde slijedi:

$$\mathrm{E}\{(\rho-\mu_\rho)^2\} = \mathrm{E}\{\xi^2\} + 2\mu_\xi + 1 - \mu_\xi^2 - 2\mu_x - 1 = \mathrm{E}\{\xi^2\} - \mu_\xi^2 = \sigma_\xi^2.$$

Dakle, slučajne promjenljive ξ i ρ imaju istu varijansu. Srednja vrijednost promjenljive ξ je jednaka:

$$\mu_\xi = 0.1\cdot 2 + 0.3\cdot 4 + 0.4\cdot 6 + 0.1\cdot 8 + 0.1\cdot 10 = 5.6,$$

dok je srednja kvadratna vrijednost:

$$\mathrm{E}\{\xi^2\} = 0.1\cdot 4 + 0.3\cdot 16 + 0.4\cdot 36 + 0.1\cdot 64 + 0.1\cdot 100 = 36.$$

Sada dobijamo da je varijansa slučajnih promjenljivih ξ i ρ jednaka:

$$\sigma_\xi^2 = 36 - 5.6^2 = 4.64.$$

Odredite zavisnost srednje vrijednosti i varijanse slučajne promjenljive ρ od ovih parametara za slučajnu promjenljivu ξ, ako je veza između ovih veličina linearna:

$$\rho = a\xi + b.$$

1.6. Slučajna promjenljiva sa jednakim vjerovatnoćama uzima vrijednosti –1, 0, 1. Odrediti srednju vrijednost i varijansu slučajnih promjenljivih: $\eta_1 = 1 - 2\xi^2$ i $\eta_2 = a + \xi^2$.

Rješenje: Slučajne promjenljive η_1 i η_2 uzimaju se s vjerovatnoćama navedenim u Tabeli I.8, a što je dalje pojednostavljeno u Tabeli I.9.

Srednja vrijednost navedenih slučajnih promjenljivih je:

$$\mu_{\eta_1} = -\frac{2}{3} + \frac{1}{3} = -\frac{1}{3} \qquad \mu_{\eta_2} = \frac{2(a+1)}{3} + \frac{a}{3} = a + \frac{2}{3}.$$

ξ	-1	0	-1
η_1	-1	1	-1
η_2	a+1	A	a+1
p	1/3	1/3	1/3

Tabela I.8. Tabela sa slučajnim vjerovatnoćama i vrijednostima slučajnih promjenljivih ξ, η_1, η_2

η_1	-1	1
η_2	a+1	A
P	2/3	1/3

Tabela I.9. Pojednostavljena Tabela I.8

Srednje kvadratne vrijednosti ovih promjenljivih su:

$$E\{\eta_1^2\} = \frac{2}{3} + \frac{1}{3} = 1 \qquad E\{\eta_2^2\} = \frac{2(a+1)^2}{3} + \frac{a^2}{3} = a^2 + \frac{4a}{3} + \frac{2}{3}.$$

Sada su varijanse jednake:

$$\sigma_{h_1}^2 = E\{\eta_1^2\} - \mu_{\eta_1}^2 = 1 - \frac{1}{9} = \frac{8}{9} \qquad \sigma_{h_2}^2 = E\{\eta_2^2\} - \mu_{\eta_2}^2 = \frac{2}{3} - \frac{4}{9} = \frac{2}{9}.$$

1.7. Prenose se dvije poruke. Slučajna promjenljiva ξ uzima vrijednost 1 ako je prva poruka uspješno prenesena i 0 ako nije. Slučajna promjenljiva η se na isti način odnosi prema drugoj poruci. Ako je vjerovatnoća uspješnog prenosa prve poruke p_1, a druge poruke p_2, odrediti združenu raspodjelu slučajnih promjenljivih ξ i η.

Rješenje: Združene vjerovatnoće se određuju jednostavno:

$$P\{\xi = 1, \eta = 1\} = p_1 p_2 \qquad P\{\xi = 1, \eta = 0\} = p_1(1 - p_2)$$

$$P\{\xi = 0, \eta = 1\} = (1 - p_1) p_2 \qquad P\{\xi = 0, \eta = 0\} = (1 - p_1)(1 - p_2).$$

1.8. Data je uslovna gustina raspodjele $p_{\xi_1\xi_2|\xi_3\xi_4}(x_1, x_2 \mid x_3, x_4)$. Na osnovu nje treba odrediti $p_{\xi_1|\xi_3\xi_4}(x_1 \mid x_3, x_4)$. Ovdje važi pravilo koje treba dokazati:

$$p_{\xi_1|\xi_3\xi_4}(x_1 \mid x_3, x_4) = \int_{-\infty}^{\infty} p_{\xi_1\xi_2|\xi_3\xi_4}(x_1, x_2 \mid x_3, x_4) dx_2.$$

Rješenje: Događaj $\xi_1\xi_2 \mid \xi_3\xi_4$ predstavlja uniju događaja da se dogodilo ξ_1, pod uslovom da se dogodilo $\xi_3\xi_4$, i događaja ξ_2, pod uslovom da se dogodilo $\xi_3\xi_4$. Stoga, važi odgovarajuća veza sa marginalnom vjerovatnoćom:

$$p(x) = \int_{-\infty}^{\infty} p(x, y)dy$$

gdje je x u ovom slučaju događaj $\xi_1 \mid \xi_3\xi_4$, dok je događaj y zapravo $\xi_2 \mid \xi_3\xi_4$, čime smo dokazali predmetnu tvrdnju.

1.9. Zadata je uslovna gustina raspodjele: $p_{\xi_1\xi_2\xi_3|\xi_4\xi_5\xi_6}$. Traži se uslovna gustina $p_{\xi_1\xi_2\xi_3|\xi_4\xi_6}$. Ovdje važi pravilo koje treba dokazati:

$$p_{\xi_1\xi_2\xi_3|\xi_4\xi_6} = \int_{-\infty}^{\infty} p_{\xi_1\xi_2\xi_3|\xi_4\xi_5\xi_6}\, p_{\xi_5|\xi_4\xi_6}\, dx_5.$$

Rješenje: Predmetni izraz možemo preglednije prikazati kao:

$$p(a \mid c) = \int_{-\infty}^{\infty} p(a \mid bc)\, p(b \mid c)db,$$

gdje je $a = \xi_1\xi_2\xi_3$, $c = \xi_4\xi_6$ i $b = \xi_5$. Ako sada uzmemo da se dogodilo, ovo se svodi na Bajesovo pravilo za kontinualne slučajne promjenljive.

1.10. Sistem prenosi tri poruke A, B i C koje se mogu izabrati sa vjerovatnoćama p_A, p_B, p_C, respektivno. Važi $p_A + p_B + p_C = 1$. Vjerovatnoća da je primljena poruka Q ako je poslata poruka R je $P_{Q|R}$ (npr. ako je poslato A, a primljeno B, to je $P_{B|A}$). Odrediti vjerovatnoće primljenih simbola. Odrediti uslovne vjerovatnoće da ako je primljen neki simbol da je taj simbol i poslat.

Rješenje: Simbol A je primljen ako je poslato A a primljeno A, poslato B − primljeno A, poslato C a primljeno A, što se može zapisati kao:

$$p(A) = p_A P_{A|A} + p_B P_{A|B} + p_C P_{A|C}$$
$$p(B) = p_A P_{B|A} + p_B P_{B|B} + p_C P_{B|C}$$
$$p(C) = p_A P_{C|A} + p_B P_{C|B} + p_C P_{C|C},$$

odnosno u matričnom obliku:

$$\begin{bmatrix} p(A) \\ p(B) \\ p(C) \end{bmatrix} = \begin{bmatrix} P_{A|A} & P_{A|B} & P_{A|C} \\ P_{B|A} & P_{B|B} & P_{B|C} \\ p_{C|A} & P_{C|B} & P_{C|C} \end{bmatrix} \begin{bmatrix} p_A \\ p_B \\ p_C \end{bmatrix}.$$

Uslovna vjerovatnoća da je poslato A, ako je A primljeno, jeste:

$$P(A \mid A) = \frac{P(A, A)}{p(A)} = \frac{p_A P_{A|A}}{p_A P_{A|A} + p_B P_{A|B} + p_C P_{A|C}},$$

gdje je $P(A, A)$ združena vjerovatnoća da je poslato A i primljeno A.

1.11. Diskretna slučajna promjenljiva ξ uzima vrijednosti: {1, 2, 3, 4, 5}, sa vjerovatnoćama {0.1, 0.3, 0.4, 0.1, 0.1}, respektivno. Odrediti raspodjelu i očekivanu vrijednost za slučajnu promjenljivu $\eta = (\xi - 3)^2$. Odrediti združene raspodjele, kao i odgovarajuće uslovne vjerovatnoće ove dvije slučajne promjenljive.

Rješenje: Formirajmo tabelu sa vrijednostima promjenljivih ξ i η i sa odgovarajućim vjerovatnoćama (Tabela I.10). Pojednostavimo tabelu za događaj η (Tabela I.11).

Združena raspodjela ove dvije slučajne promjenljive može se prikazati Tabelom I.12, dok su uslovne vjerovatnoće prikazane u Tabeli I.13.

1.12. Odredili ste srednju vrijednost i varijansu neke slučajne poruke na osnovu procijenjenih vjerovatnoća pojedinih događaja: $p_i, i = 1,...,N$. Međutim, stvarne vrijednosti vjerovatnoća pojedinih događaja su $p_i + e_i, i = 1,...,N$. Koliko je srednje odstupanje i srednje kvadratno odstupanje određene veličine u odnosu na tačnu veličinu? Pretpostaviti da je srednja kvadratna vrijednost $\mathrm{E}\{e_i^2\} = \Delta^2$.

ξ	1	2	3	4	5
η	4	1	0	1	4
p	0.1	0.3	0.4	0.1	0.1

Tabela I.10. Vjerovatnoće događaja (alfabeta) ξ i η

η	0	1	4
p	0.4	0.4	0.2

Tabela I.11. Pojednostavljena vjerovatnoća događaja η

$p(\xi,\eta)$	ξ=1	ξ=2	ξ=3	ξ=4	ξ=5	$p(\eta)$
η=0	0	0	0.4	0	0	0.4
η=1	0	0.3	0	0.1	0	0.4
η=4	0.1	0	0	0	0.1	0.2
$p(\xi)$	0.1	0.3	0.4	0.1	0.1	

Tabela I.12. Združena i marginalne vjerovatnoće događaja ξ i η

$p(\xi\|\eta)$	ξ=1	ξ=2	ξ=3	ξ=4	ξ=5
η=0	0	0	1	0	0
η=1	0	0.75	0	0.25	0
η=4	0.5	0	0	0	0.5

$p(\eta\|\xi)$	ξ=1	ξ=2	ξ=3	ξ=4	ξ=5
η=0	0	0	1	0	0
η=1	0	1	0	1	0
η=4	1	0	0	0	1

Tabela I.13. Uslovne vjerovatnoće događaja ξ i η

Rješenje: Srednje odstupanje vjerovatnoće je jednako:

$$\mathrm{E}\{p_i + e_i\} = \mathrm{E}\{p_i\} + \mathrm{E}\{e_i\} = p_i,$$

pod pretpostavkom da je $\mathrm{E}\{e_i\} = 0$, što je realno, jer suma vjerovatnoća mora biti jednaka 1, tako da i suma $p_i + e_i$ mora biti jednako 1. Ostaje da je suma grešaka jednaka 0, odnosno da je odgovarajuće matematičko očekivanje jednako 0. Zbog toga možemo reći da je procjena vjerovatnoće nekog elementa iz alfabeta u ovom slučaju nepristrasna (engl. *ubiased*). Što se tiče varijanse, lako se može pokazati da važi da je:

$$\sigma^2 = \mathrm{E}\{(p_i + e_i)^2\} - \mathrm{E}^2\{p_i^2\} = \mathrm{E}\{p_i^2\} - 2\mathrm{E}\{p_i e_i\} + \mathrm{E}\{e_i^2\} - \mathrm{E}\{p_i^2\} =$$
$$= -2\mathrm{E}\{p_i e_i\} + \Delta^2.$$

Pod pretpostavkom da su vjerovatnoće događaja p_i i greške u procjeni vjerovatnoće toga događaja nezavisni, onda slijedi:

$$\mathrm{E}\{p_i e_i\} = \mathrm{E}\{p_i\}\mathrm{E}\{e_i\} = p_i \mathrm{E}\{e_i\} = p_i \cdot 0 = 0,$$

pa dobijamo:

$$\sigma^2 = \Delta^2.$$

1.13. Na pravcu kretanja automobila nalaze se 3 semafora. Vjerovatnoća zaustavljanja automobila na prvom semaforu je 0.4, na drugom 0.6 i na trećem 0.5. Pretpostaviti da semafori rade nezavisno jedan od drugog. Naći raspodjelu slučajne promjenljive X koja predstavlja broj semafora koje je vozač automobila prošao do prvog zaustavljanja.

Rješenje: Riječ je o relativno jednostavnom problemu. Vjerovatnoća da automobil nije prošao nijedan semafor je 0.4:

$$P(X = 0) = 0.4.$$

Vjerovatnoća da je prošao jedan semafor je:

$$P(X = 1) = (1 - P(X = 0)) \cdot 0.6 = 0.36.$$

Vjerovatnoća da je prošao dva semafora je:

$$P(X = 2) = (1 - P(X = 0) - P(X = 1))\ 0.5 = 0.12.$$

Konačno, vjerovatnoća da je prošao sva tri semafora je:

$$P(X = 3) = (1 - P(X = 0) - P(X = 1) - P(X = 2)) = 0.12.$$

U smislu funkcije raspodjele, ovo se može zapisati kao:

$$P_{\xi}(X)=\begin{cases}0 & X<0\\ 0.4 & X\in[0,1)\\ 0.4+0.36=0.76 & X\in[1,2)\\ 0.88 & X\in[2,3)\\ 1 & X\geq 3.\end{cases}$$

Pošto se radi o diskretnoj slučajnoj promjenljivoj, u pravom smislu riječi ne baratamo funkcijom gustine raspodjele, ali je ipak možemo zapisati putem generalisane funkcije, kao:

$$p_{\xi}(X)=P_{\xi}'(X)=0.4\delta(X)+0.36\delta(X-1)+0.12\delta(X-2)+0.12\delta(X-3),$$

gdje je $\delta(X)=1$ za $X=0$ i $\delta(X)=0$, za druge vrijednosti X, diskretna Dirakova funkcija.

1.14. Radnik svaki dan ide na posao autobusom, taksijem ili biciklom. U 50% slučajeva ide autobusom, u 20% slučajeva odlučuje se za taksi, a u 30% slučajeva ide biciklom. Vjerovatnoća da će zbog kašnjenja autobusa zakasniti na posao je 0.05, u slučaju izbora taksija 0.1, a biciklom kasni sa vjerovatnoćom 0.01. Izračunati vjerovatnoću sa kojom radnik kasni na posao. Ako je radnik zakasnio na posao, koliko je vjerovatnoća da je išao biciklom?

Rješenje: Vjerovatnoća da je radnik zakasnio na posao je:

$$P_{\text{kaanjenje}}=P(a)P(k\mid a)+P(t)P(k\mid t)+P(b)P(k\mid b)=$$
$$=0.5\cdot 0.05+0.2\cdot \mathbf{0}.1+0.3\cdot 0.01=0.048,$$

gdje smo sa a, t i b označili izbor odlaska na posao automobilom, taksijem i biciklom, dok je k događaj kašnjenja na posao. Uslovnu vjerovatnoću da je radnik išao biciklom, ako se kašnjenje dogodilo, dobijamo kao:

$$P(b\mid k)=\frac{P(b)P(k\mid b)}{P_{\text{kaanjenje}}}=\frac{0.003}{0.048}=\frac{1}{16}.$$

1.15. Tri komunikaciona kanala se koriste za prenos podataka i to čine na sljedeći način: prvi kanal prenosi 10 bita, od čega su 4 neispravna; drugi kanal prenosi 6 bita, od čega je 1 neispravan; treći kanal prenosi 8 bita, od čega su 3 neispravna. Iz slučajno odabranog kanala nasumice se odabira jedan bit. Odrediti vjerovatnoću da je odabrani bit neispravan. Odrediti vjerovatnoću da je odabrani bit iz drugog kanala, ako se zna da je taj bit ispravan.

Rješenje: Označimo događaje: A – „odabrani bit je neispravan“; B_i – „odabrani bit potiče iz komunikacionog kanala $i\in\{1,2,3\}$“. Pošto je $\{B_1, B_2, B_3\}$ potpun sistem događaja, očigledno je:

$$P(B_i)=\frac{1}{3}, i\in\{1,2,3\}.$$

Imajte u vidu da su odabiri kanala međusobno nezavisni događaji. Dalje važi:

$$P(A\mid B_1)=\frac{4}{10}=\frac{2}{5},\quad P(A\mid B_2)=\frac{1}{6},\quad P(A\mid B_3)=\frac{3}{8}.$$

Na osnovu formule totalne vjerovatnoće, možemo pisati:

$$P(A)=\sum_{i=1}^{3}P(B_i)P(A\mid B_i)=\frac{1}{3}\cdot\frac{2}{5}+\frac{1}{3}\cdot\frac{1}{6}+\frac{1}{3}\cdot\frac{3}{8}=\frac{113}{360}\approx 0.3139.$$

Neka je $\overline{A}$ **komplement** događaja A, odnosno „odabrani bit je ispravan“, dobijamo:

$$P(\overline{A})=1-P(A)\approx 1-0.3139=0.6861,$$

$$P(\overline{A}\mid B_2)=1-P(A\mid B_2)=\frac{5}{6}.$$

Primjenom Bajesove jednakosti dobijamo da je:

$$P(B_2\mid\overline{A})=\frac{P(B_2)P(\overline{A}\mid B_2)}{P(\overline{A})}\approx\frac{\frac{1}{3}\cdot\frac{5}{6}}{0.6861}=0.4049.$$

1.16. Firma ima na raspolaganju 6 telefonskih linija. Neka je X broj linija zauzetih u određenom trenutku. Zakon raspodjele za X je dat sa:

$$X:\begin{matrix}x_i\\p(x_i)\end{matrix}\begin{pmatrix}0 & 1 & 2 & 3 & 4 & 5 & 6\\ 0.1 & 0.15 & 0.2 & 0.25 & 0.2 & 0.06 & 0.04\end{pmatrix}.$$

Izračunati vjerovatnoće sljedećih događaja: A – „bar 3 linije su zauzete“, B – „manje od 2 linije su zauzete“, C – „najmanje 4 linije nisu zauzete“, D – „zauzeto je između 2 i 5 linija“.

Rješenje: Tražene vjerovatnoće su:

$$P(A)=P(X\geq 3)=P(X=3)+P(X=4)+P(X=5)+P(X=6)=$$
$$=0.25+0.2+0.06+0.04=0.55$$

$$P(B)=P(X<2)=P(X=0)+P(X=1)=0.1+0.15=0.25$$

$$P(C)=P(X\leq 2)=P(X=0)+P(X=1)+P(X=2)=0.1+0.15+0.2=0.45$$

$$P(D)=P(2\leq X\leq 5)=P(X=2)+P(X=3)+P(X=4)+P(X=5)=$$
$$=0.2+0.25+0.2+0.06=0.71.$$

1.17. Data je funkcija raspodjele slučajne promjenljive:

$$P_{\xi}(x)=\begin{cases}0 & x<0\\ 0.1 & 0\le x<1\\ 0.25 & 1\le x<2\\ 0.45 & 2\le x<3\\ 0.7 & 3\le x<4\\ 0.9 & 4\le x<5\\ 0.96 & 5\le x<6\\ 1 & 6\le x.\end{cases}.$$

Odrediti funkciju gustine raspodjele ove slučajne promjenljive. Odrediti srednju vrijednost i varijansu slučajne promjenljive.

Rješenje: Korišćenjem generalizovane Dirakove funkcije, funkciju gustine raspodjele možemo zapisati:

$$p_{\xi}(x)=P_{\xi}'(x)=0.1\delta(x)+0.15\delta(x-1)+0.2\delta(x-2)+0.25\delta(x-3)+$$
$$+0.2\delta(x-4)+0.06\delta(x-5)+0.04\delta(x-6).$$

Srednju vrijednost i varijansu sada možemo da sračunamo na različite načine, a ovdje će biti korištena formula za kontinualne slučajne promjenljive (premda se radi o diskretnim, ali iz činjenice da koristimo generalisanu funkciju, možemo koristiti predmetni luksuz):

$$\mu_x=E\{x\}=\int_{-\infty}^{\infty}xp_{\xi}(x)dx=0.1+2\cdot0.2+3\cdot0.25+4\cdot0.2+5\cdot0.06+6\cdot0.04=$$
$$=2.59$$
$$\sigma_x^2=E\{(x-\mu_x)^2\}=E\{x^2\}-\mu_x^2=$$
$$=0.1+4\cdot0.2+9\cdot0.25+16\cdot0.2+25\cdot0.06+36\cdot0.04-\mu_x^2=$$
$$=2.5819.$$

1.18. Putem Euklidskog algoritma dokazati da se razlomak:

$$\frac{N+2}{2N+5}$$

ne može skratiti ni za jedan prirodan broj N.

Rješenje: Prvo uočimo da za svaki prirodni broj N važi da je $2N+5>N+2$. Dakle, važi:

$$\gcd(2N+5,N+2)=\gcd(N+2,1),$$

jer je količnik $2N+5$ sa $N+2$ broj 2, a ostatak je 1. Očigledno je najveći zajednički djelilac ova dva broja 1, što znači da ne postoji cijeli broj kojim se mogu skratiti brojevi u imeniocu i brojiocu.

1.19. Euklidskim algoritmom odrediti najmanji prirodni broj N za koji se razlomak može skratiti i odrediti vrijednost razlomka u datom slučaju:

$$\frac{15N-7}{22N-5}.$$

Rješenje: Ponovimo postupak opisan u prethodnom zadatku. Napominjemo da je potrebno u svakom koraku provjeriti koji od dobijenih izraza je veći, jer se eventualno može značajno usložniti dati postupak:

$$\gcd(22N-5,15N-7)=\gcd(15N-7,7N+2)=\gcd(7N+2,N-11)=\gcd(N-11,79).$$

Napominjemo da je u prvom koraku količnik 1, u drugom 2, a u trećem 7. Sada vidimo da za $N=79+11=90$ dobijamo da je najmanji zajednički djelilac imenioca i brojioca izraza 79, pa se u tom slučaju imenilac i brojilac mogu skratiti za 79:

$$\frac{15\cdot 90-7}{22\cdot 90-5}=\frac{1343}{1975}=\frac{17}{25}.$$

1.20. Dokazati da skup $P=\{0, 1, 2, \ldots, p-1\}$ s operacijama sabiranja i množenja, definisanim po modulu p, predstavlja polje ako je p prost broj. Pokazati da ovakav skup s operacijama nije polje ako je broj p složen.

Rješenje: Prvo je potrebno dokazati da je predmetni skup polje za p prost broj. Nije teško uočiti da je jedina problematična osobina polja, koja eventualno ne može biti zadovoljena, osobina 8, odnosno postojanje inverznog elementa po množenju (multiplikativnog inverza):

Za svako nenulto $x \in X$ koje $x \neq 0$, postoji $x^{-1} \in X$, tako da $x^{-1}\,x = x\,x^{-1} = 1$.

Postupak dokazivanja za inverz sa lijeve i sa desne strane je istovjetan, tako da ćemo se zadržati samo na dokazivanju da postoji broj y takav da je $x \cdot y = 1$. Kako je

$$\mathrm{mod}(xy,p)=1,$$

to znači da je $xy=kp+1$, gdje je k cijeli broj. Ovo dalje znači da:

$$xy+rp=1$$

u datoj modularnoj algebri (ovdje je $r=-k$). Ako pogledamo sada ovaj izraz, on predstavlja formulaciju iz Euklidskog algoritma, primijenjenog na određivanje najvećeg zajedničkog djelioca za dva broja (neka su to ovdje x i y). Odavde slijedi

da je taj najveći zajednički djelilac jednak 1 (što mora biti zadovoljeno ako je p prost broj za svaki element skupa), pa smo sigurni da u datom skupu za svako $x \neq 0$ postoji multiplikativni inverz u ovom skupu.

Situacija nije takva kod skupova kod kojih je p složen broj, odnosno kod kojih se p može prikazati kao proizvod dva (prosta ili složena broja) broja $p = p_1 p_2$. Kod ovakvog skupa neki elementi nemaju multiplikativni inverz. Na primjer, ako je $p = 8 = 2 \cdot 4$, za $x = 3$ postoji multiplikativni inverz $y = 3$, ali za $x = 2$ ne postoji multiplikativni inverz jer nijedan broj pomnožen sa x ne daje neparan broj, a samim tim ni neparni ostatak pri dijeljenju sa 8, a to dalje znači da ne može ni ostatak pri dijeljenju sa 8 biti 1. Kako ne postoji multiplikativni inverz za sve nenulte elementi predmetni skup nije polje.

1.21. Dokazati sljedeće stavove za pojedine vrste, u tabeli koja se dobija u postupku Euklidskog algoritma:

(a) $r_{k-1}v_k - r_k v_{k-1} = \pm a \quad u_{k-1}v_k - u_k v_{k-1} = \pm 1 \qquad k \geq 0$

(b) $u_k a + v_k b = r_k \qquad \forall k$

(c) $\deg(u_k) + \deg(r_{k-1}) = \deg(b) \qquad k \geq 1$

(d) $\deg(v_k) + \deg(r_{k-1}) = \deg(a) \qquad k \geq 0.$

Rješenje: Započnimo od prvog stava. Dokaz obavimo matematičkom indukcijom. Za $k = 0$ slijedi da treba da dokažemo da je

$$v_0 r_{-1} - v_{-1} r_0 = a.$$

Kako je $v_0 = 1$, $v_{-1} = 0$, $r_{-1} = a$ i $r_0 = b$, time je navedena činjenica dokazana. Sada, po pravilima indukcije, treba dokazati da, pod uslovom da relacija važi za k:

$$v_k r_{k-1} - v_{k-1} r_k = (-1)^k a,$$

važi i za $k+1$:

$$v_{k+1} r_k - v_k r_{k+1} = (-1)^{k+1} a.$$

Sada uvrstimo u ovu relaciju:

$$v_{k+1} = v_{k-1} - q_{k+1} v_k$$
$$r_{k+1} = r_{k-1} - q_{k+1} r_k,$$

pa dobijamo:

$$(v_{k-1} - q_{k+1}v_k) r_k - v_k (r_{k-1} - q_{k+1} r_k) = v_{k-1} r_k - q_{k+1} v_k r_k - v_k r_{k-1} + q_{k+1} v_k r_k =$$
$$= v_{k-1} r_k - v_k r_{k-1} = -(-1)^k a = (-1)^{k+1} a.$$

Pređimo sada na dokaz drugog stava (b):

$$u_k v_{k-1} - u_{k-1} v_k = (-1)^{k+1}.$$

Ponovo pokrenimo mehanizam matematičke indukcije, dokazujući da predmetna relacija važi za $k = 0$:

$$u_0 v_{-1} - u_{-1} v_0 = -1.$$

Uvrštavajući početne vrijednosti $u_0 = 0$, $u_{-1} = 1$, $v_0 = 1$, $v_{-1} = 0$ dobijamo $-1 = -1$ (odnosno u aritmetici po modulu 2: $-1 = 1$). Sada treba da dokažemo da ako važi gorenavedena relacija za k, onda važi i za $k+1$:

$$u_{k+1} v_k - u_k v_{k+1} = (-1)^{k+2}.$$

Uvrštavajući u ovu jednačinu pravila ažuriranja za kolone $\boldsymbol{U}$ i $\boldsymbol{V}$:

$$u_{k+1} = u_{k-1} - q_{k+1} u_k$$

$$v_{k+1} = v_{k-1} - q_{k+1} v_k,$$

dobijamo

$$\begin{aligned}(u_{k-1} - q_{k+1}u_k)v_k - u_k(v_{k-1} - q_{k+1}v_k) &= u_{k-1}v_k - u_k v_{k-1} - q_{k+1}u_k v_k + q_{k+1}u_k v_k = \\ &= u_{k-1}v_k - u_k v_{k-1} = -(-1)^{k+1} = (-1)^{k+2}\end{aligned}$$

čime je dokaz kompletiran.

Pređimo na treći stav (c). Ovdje ćemo odstupiti od dokazivanja matematičkom indukcijom, pa ćemo se prebaciti na direktni dokaz. Pođimo od sljedećeg izraza:

$$r_{k-1} u_k - r_k u_{k-1} = \pm b.$$

Stepen polinoma r_k opada s rastom k, pa važi $\deg(r_k) < \deg(r_{k-1})$, dok stepen polinoma u_i raste $\deg(u_k) > \deg(u_{k-1})$. Stoga je $\deg(r_{k-1}u_k) > \deg(r_k u_{k-1})$, pa odatle slijedi da je $\deg(r_{k-1}u_k - r_k u_{k-1}) = \deg(r_{k-1}\, u_k) = \deg(r_{k-1}) + \deg(u_k) = \deg(b)$, čime je dokaz završen.

Dokaz posljednjeg stava (pod (d)) obavlja se na isti način kao dokaz pod (c), pa ga možete obaviti sami.

1.22. Izvršiti Euklidski algoritam za polinome s binarnim koeficijentima $a(x) = x^7 + 1$ i $b(x) = x^6 + x^4 + x^2 + x + 1$.

Rješenje: Tabela I.14 sublimira korake Euklidskog algoritma. U ovom slučaju nije bilo očigledno da ćemo kao rezultat dobiti međusobno proste polinome.

k	Q	R	U	V
-1	...	x^7+1	1	0
0	...	$x^6+x^4+x^2+x+1$	0	1
1	x	$x^5+x^3+x^2+x+1$	1	X
2	x	x^3+1	x	x^2+1
3	x^2+1	X	x^3+x+1	x^4+x+1
4	x^2	1	$x^5+x^3+x^2+x$	x^6+x^3+1
5	X	0	$x^6+x^4+x^2+x+1$	x^7+1

Tabela I.14. Euklidski algoritam za binarne polinome

K	Q	R	U	V
-1	...	x^{10}	1	0
0	...	x^4+x^3+1	0	1
1	$x^6+x^5+x^4+x^3+x$	x^3+x	1	$x^6+x^5+x^4+x^3+x$
2	$x+1$	x^2+x+1	$x+1$	$x^7+x^3+x^2+x+1$
3	$x+1$	$x+1$	x^2	$x^8+x^7+x^6+x^5+x^3+x+1$
4	x	1	x^3+x+1	$x^9+x^8+x^6+x^4+x^3+1$
5	$x+1$	0	x^4+x^3+1	x^{10}

Tabela I.15. Euklidski algoritam za binarne polinome

1.23. Za polinome $a(x)=x^{10}$ i $b(x)=x^4+x^3+1$ obavite Euklidski algoritam.

Rješenje: Izvršimo Euklidski algoritam sa navedenim polinomima koracima datim u Tabeli I.15. U ovom slučaju je inače očigledno da su polinomi uzajamno prosti.

1.24. Dokazati da ako je binarni polinom:

$$\mathbf{c}(x)=\sum_{k=0}^{n}a_k x^k$$

prost, tada je prost i polinom:

$$\overline{\mathbf{c}}(x)=\sum_{k=0}^{n}a_k x^{n-k}.$$

Rješenje: Dokaz se može provesti prilično intuitivno. Prvo je potrebno dokazati da ako imamo proizvod dva polinoma $\mathbf{a}(x)\mathbf{b}(x)$ koji je jednak nekom polinomu:

$$\mathbf{a}(x)\mathbf{b}(x)=\sum_{p=0}^{P}\alpha_p x^p$$

tada će proizvod polinoma s obrnutim koeficijentima (označimo ih sa $\overline{\mathbf{a}}(x)$ i $\overline{\mathbf{b}}(x)$) biti:

$$\overline{\mathbf{a}}(x)\overline{\mathbf{b}}(x) = \sum_{p=0}^{P} \alpha_{P-p} x^p .$$

Sada možemo zaključiti da ako $\sum_{p=0}^{P} \alpha_p x^p$ nije prost, odnosno ako ima djelioce, onda ni polinom $\sum_{p=0}^{P} \alpha_{P-p} x^p$ nije prost.

1.25. Dokazati da, ako binarni polinom ima nenulte koeficijente samo uz parne stepene, ne može biti prost.

Rješenje: Uzmimo da imamo polinom sa samo parnim koeficijentima:

$$\mathbf{c}(x) = \sum_{p=0}^{P} a_p x^{2p} .$$

Posmatrajmo sada polinom

$$\mathbf{q}(x) = \sum_{p=0}^{P} a_p x^p .$$

Izvršimo kvadriranje ovog polinoma:

$$\mathbf{q}^2(x) = \sum_{p_1=0}^{P} \sum_{p_2=0}^{P} a_{p_1} a_{p_2} x^{p_1+p_2} =$$

$$= \sum_{p=0}^{P} a_p^2 x^{2p} + 2 \sum_{p_1=0}^{P} \sum_{\substack{p_2=0 \\ p_1 \neq p_2}}^{P} a_{p_1} a_{p_2} x^{p_1+p_2} .$$

Već smo vidjeli da važi da je stepen binarnog broja jednak samom tom broju, te da je druga suma u predmetnom izrazu jednaka 0. Stoga, možemo zaključiti da je $\mathbf{q}^2(x) = \mathbf{c}(x^2)$, a samim tim da je $\mathbf{c}(x)$ djeljivo sa $\mathbf{q}(x)$, pa $\mathbf{c}(x)$ nije prost polinom.

I.7.2 Softverska realizacija

Ideja je da na kraju svakog poglavlja damo nekoliko osnovnih pouka kako se softverski paket MATLAB ili Octave (pošto imaju gotovo identičnu sintaksu) mogu koristiti za potrebe testiranja pojedinih koncepata teorije informacija i kodova. Ni po čemu ovaj materijal nije sveobuhvatan, niti na optimalan i sistematičan način obrađuje koncepte, već je samo putokaz kako se mogu simulirati i demonstrirati procesi i algoritmi u ovoj oblasti. Ponekad to radimo putem najprimitivnijih naredbi (bez obzira što postoje kvalitetnije), ponekad koristimo naredbe iz pojedinih MATLAB-ovih biblioteka, a kada ni jedno ni drugo nije dovoljno, idemo u nekoliko slučajeva i korak naprijed u ovoj tematici, koristivši programe dostupne na razmjeni

putem interneta. MATLAB se razvija i sve više izlazi u susret korisnicima u ovoj oblasti, tako da gotovo svaka nova verzija ovog paketa smanjuje potrebu da se za osnovne postupke i algoritme vrši pretraga po internetu.

A. Generišimo sekvencu od deset slučajnih simbola binarnog alfabeta:

```
>> rand(1,10)>0.5
```

Naime, ova funkcija generiše vektor od 10 simbola sa jedinicama na pozicijama gdje je rand(1,10) veće od 0.5. Kako rand uzima slučajne vrijednosti u intervalu (0,1) s uniformnom vjerovatnoćom, ovo, recimo, simulira slanje binarne poruke kod koje se s istom vjerovatnoćom šalju obje vrijednosti bita.

B. Generišimo 1000 sekvenci po 10 slučajnih bita i prebrojimo koliko se jedinica pojavljuje u svakoj sekvenci:

```
>> R=[];
>> for k=1:1000
R(k)=sum(rand(1,10)>0.5);
end
```

Očekujemo da ćemo dobiti 5 jedinica. Međutim, u kratkim sekvencama to ne mora biti tako, pa sljedećim naredbama (moguća su odstupanja u zavisnosti od generisanja slučajnih brojeva) dobijamo:

```
>> for r=0:10
length(find(R==r))
end
1   6   27   114   213   237   223   118   46  14  1
```

Vidimo da se u ovoj kratkoj sekvenci dešava da dobijemo slučaj sa deset nula ili deset jedinica! U 237 slučajeva (od 1000) dobili smo 5 nula i 5 jedinica, a u još 436 slučajeva, 4 ili 6 jedinica. Rezultati svih eksperimenata sa generatorima slučajnih brojeva mogu varirati od realizacije do realizacije. Ponovimo eksperiment sa sekvencom koja ima 1000 nula ili jedinica:

```
>> for k=1:1000
R(k)=sum(rand(1,1000)>0.5);
end
```

Pogledajmo sada:

```
>> find(R<450)
ans =[]
>> find(R>550)
ans =[]
```

Dakle, sada smo dobili znatno bolju pravilnost, odnosno svih hiljadu slučajnih događaja u našoj realizaciji produkuje sekvence sa 450 do 550 jedinica. U velikom broju ponavljanja dobijamo bolju pravilnost nego u malom broju ponavljanja.

C. Generišimo binarnu sekvencu sa 10 simbola sa 60% jedinica:

```
>> R=rand(1,10)>0.4;
```

D. Demonstrirati situaciju kada je prvi bit nula ili jedinica s istom vjerovatnoćom, a zatim se prethodno stanje zadržava s vjerovatnoćom 0.7, a prelazi se u drugo stanje s vjerovatnoćom 0.3. Ovo se može programirati na različite načine, a dolje je demonstrirano kreiranje sekvence od 10000 bita koji su dobijeni na prethodno opisani način:

```
>> N=10000; R=zeros(1,N);
>> R(1)=rand>0.5;
>> for k=2:N,R(k)=rem(R(k-1)+(rand>0.7),2);end
```

Ovdje koristimo zgodnu funkciju rem, koju ćemo često upotrebljavati, a koja nam daje ostatak pri dijeljenju (u ovom slučaju) sa dva ili ekskluzivno ili operaciju. Ako je (rand>0.7) jednak nuli, a to će se dogoditi u 70% situacija (s vjerovatnoćom 0.7), ostatak pri dijeljenju će biti jednak vrijednosti prvog argumenta R(k-1), odnosno ostajemo u stanju R(k-1) (prethodnom stanju), bilo da je nula ili jedan. Ako je (rand>0.7) jednak 1, dolazi do promjene stanja jer je rem(0+1,2) jednako 1, odnosno rem(1+1,2) jednako 0, bilo da je R(k-1) nula ili jedinica.

E. Posmatrajmo još jednu jednostavnu simulaciju. Želimo da dobijemo slučajni proces koji s jednakom vjerovatnoćom daje vrijednosti iz skupa (alfabeta) {0, 1, 2}. To se može postići na sljedeći način:

```
>> R=floor(3*rand(1,1000))
```

Proizvod 3*rand(1,1000) daje uniformno raspoređene slučajne brojeve na intervalu [0,3). Funkcija floor ih zaokružuje naniže, tako da dobijamo samo 0, 1 i 2, i to sa jednakim vjerovatnoćama. MATLAB posjeduje i naredbu kojom slučajne cijele brojeve kreira bez potrebe da se množi cijeli broj s uniformnim šumom. Tako naredba:

```
>> randi([0 4],[1 7])
```

kreira niz od 7 elemenata (matricu dimenzija 1×7 specificiranu kao drugi argument naredbe) na alfabetu, u intervalu od 0 do 4 (prvi argument naredbe).

F. Ako postoji potreba za kreiranjem slučajnog procesa – šuma, pored naredbe rand, možete koristiti naredbu randn za formiranje Gausovog slučajnog šuma sa nultom srednjom vrijednošću i varijansom 1. Ako nam treba šum sa srednjom

vrijednošću 1, dodaćemo na rezultat naredbe rand vrijednost 1, a ako nam, na primjer, treba šum varijanse s^2, pomnožićemo ga sa s. Na primjer, matrica dimenzija 100×100 koja predstavlja Gausov šum sa varijansom 4 i srednjom vrijednošću 1, dobija se kao:

```
>> cc=2*(randn([100 100]))+1
```

Postoji i mogućnost da se kreiraju slučajni procesi sa drugim karakteristima, uz malo programiranja ili korišćenjem drugih ugrađenih naredbi.

G. Polinome možemo pomnožiti naredbom conv. Na primjer, polinome $\mathbf{a}(x)=x+1$ i $\mathbf{b}(x)=x^2+2x+3$ možemo prikazati kao vektore [1 1] i [1 2 3], respektivno. Njihov proizvod je $\mathbf{c}(x)=\mathbf{a}(x)\mathbf{b}(x)=x^3+3x^2+5x+3$, što se u MATLAB-u dobija sa:

```
>> conv([1 1],[1 2 3])
   1    3    5    3
```

U slučaju da nam treba polinom u nekom vektorskom polju, na primjer, onome sa ternarnim simbolima, nakon množenja ili sabiranja polinoma treba sračunati ostatak pri dijeljenju s prostim brojem koji predstavlja broj elemenata u polju. Na primjer, za ternarni alfabet $\{0, 1, 2\}$ sa polinomima x^2+2x+1 i $x+2$, proizvod polinoma se može sračunati kao:

```
>> rem(conv([1 2 1],[1 2]),3)
   1    1    2    2
```

odnosno rezultujući polinom je: x^3+x^2+2x+2.

H. Nešto je problematičnije dijeljenje polinoma. U osnovi se za polinome sa koeficijentima iz skupa realnih brojeva može koristiti jednostavno funkcija deconv:

```
>> [z,c]=deconv([1 2 3 1],[1 2])
z =
   1    0    3
c =
   0    0    0   -5
```

Ovom narednom smo odredili količnik polinoma x^3+2x^2+3x+1 sa i $x+2$ (rezultat je x^2+3, odnosno [1 0 3] u MATLAB notaciji), dok je ostatak c (u ovom slučaju -5). Početne nule kod ostatka postoje da bi važila u MATLAB-u relacija broj=conv(z,imen)+c (gdje su broj i imen brojilac i imenilac razlomka). Eliminacija prednjačećih nula se dobija sa:

```
>> c(min(find(c~=0):end))
```

jer funkcija find daje indekse koji zadovoljavaju logički uslov (ovdje $c \neq 0$), a min daje minimalni takav indeks, pa sve do kraja ovoga vektora (end).

I. Primjena funkcije deconv na dijeljenje polinoma u konačnim poljima mora biti sa dodatnom dozom opreza. Formirajmo dva polinoma, slučajno, na intervalu 0 do 2 (ternarni simboli), jedan četvrtog, a drugi drugog stepena (zbog slučajnosti, neki rezultati će varirati u svakom pokušaju izvršavanja naredbi):

```
>> a=randi(3,1,5)-1
a =
    2    1    0    1    0
>> b=randi(3,1,3)-1
b =
    1    1    2
```

U predmetnom obliku funkcija randi slučajno generiše brojeve od 1 do 3, pa smo zbog toga izvršili umanjivanje za 1 u oba iskaza. Odradimo zatim dijeljenje polinoma:

```
>> [q,r]=deconv(a,b)
q =
    2   -1   -3
r =
    0    0    0    6    6
```

Sada vidimo da računanje ostatka sa 3 neće dati pogodne rezultate zbog negativnih rezultata koje imamo. U predmetnom slučaju ćemo to sanirati prostim sabiranjem dobijenog rezultata sa 3 i primjenom operacije računanja ostatka dijeljenja, ali u drugim slučajevima može zahtijevati nešto više koda:

```
>> q1=rem(q+3,3)
q1 =
    2    2    0
>> r1=rem(r+3,3)
r1 =
    0    0    0    0    0
```

Provjeru operacije možemo izvršiti sa:

```
>> rem(conv(q1,b)+r1,3)
ans =
    2    1    0    1    0
```

čime smo dobili polazni polinom a.

POGLAVLJE II

ENTROPIJA

II. ENTROPIJA

Drugo poglavlje je sastavljeno iz dva dijela. U prvom dijelu se upoznajemo sa vjerovatno najvažnijim pojmom u ovoj knjizi a to je ***entropija*** – mjera količine informacija. S entropijom uvodimo i druge mjere koje su neophodne: uslovnu entropiju, međusobnu informaciju, združenu entropiju, kao i veze koje postoje između ovih veličina. U drugom dijelu poglavlja uvodimo pojam ***Markovljevih sistema*** (ili izvora informacije). Riječ je o elegantnom načinu za predstavljanje izvora informacija kod kojih postoji međusobna veza između pojedinih simbola. Lako je uočiti da ako smo izgovorili dio neke riječi, nastavak te riječi nije nezavisan od onoga što je prethodilo. Konačno, poglavlje zaključujemo nizom urađenih primjera i softverskom realizacijom nekih od uvedenih koncepata.

II.1 Mjera količine informacije

II.1.1 Osnovni pojmovi

Vjerovatno smo se već upoznali s pojmom entropije iz fizike. Ovaj pojam se koristi u oblasti termodinamike da ukaže na neuređenost posmatranog sistema. U teoriji informacija, strogo matematički, entropija je mjera neodređenosti slučajne promjenljive ili alfabeta. Mi ćemo je definisati na sljedeći način.

Definicija II.1: Posmatrajmo konačni skup/alfabet X čiji su elementi $X = \{x_1, x_2, \ldots, x_N\}$ i neka je poznata vjerovatnoća pojavljivanja svakog elementa skupa X: $p_i = p(x_i)$, $i = 1, 2, ..., N$. Tada je entropija skupa/alfabeta X (ili neodređenost alfabeta X) $H(X)$ data kao:

$$H(X) = -\sum_{i=1}^{N} p(x_i) \log p(x_i) = -\sum_{i=1}^{N} p_i \log p_i .$$

Konvencionalno se usvaja da je $0\log 0 = 0$, te da je osnova logaritma veća od 1. Po pravilu, uzimaćemo da je osnova logaritma 2, pa će se entropija izražavati u bitima (u međunarodnom sistemu mjera SI, entropija se za osnovu logaritma dva izražava u Šenonima – *Sh*, po tvorcu teorije informacija i kodova). U slučaju da se za osnovu logaritma uzme 10, govorimo o entropiji mjerenoj u ditovima, dok je entropija koja se dobija za osnovu *e* (u slučaju korišćenja prirodnog logaritma ln) mjerena u natovima.

Na primjer, ako je entropija sračunata za logaritam s osnovom *a*, može se jednostavno dobiti vrijednost entropije za logaritam koji je uzet s osnovom *b*, kao:

$$H_b(X) = -\sum_{i=1}^{N} p_i \log_b p_i = -\sum_{i=1}^{N} p_i \frac{\log_a p_i}{\log_a b} = -\frac{1}{\log_a b}\sum_{i=1}^{N} p_i \log_a p_i = \frac{H_a(X)}{\log_a b}.$$

Na primjer, ako imamo entropiju sračunatu u bitima s osnovom 2, ona je u ditovima za osnovu 10 jednaka:

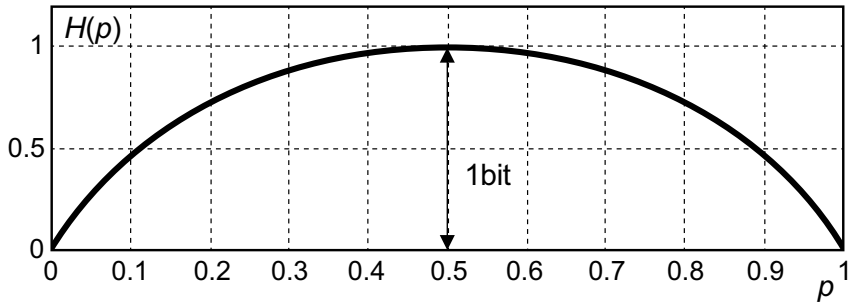
$$H_{10}(X)[\text{dit}] = \frac{H_2(X)}{\log_2 10}[\text{bit}] \approx \frac{H_2(X)}{3.322}[\text{bit}].$$

Entropija je nenegativna veličina. Osobina nenegativnosti se lako dokazuje. Entropija je suma vrijednosti oblika $-p_i \log p_i$. Vjerovatnoće su nenegativne veličine, a pošto su u granicama od [0,1], to važi $\log p_i \leq 0$, pa su članovi $p_i \log p_i \leq 0$, te sumiranjem ovih članova, uz množenje sa -1, zasigurno daju nenegativan rezultat. Postavlja se pitanje: kako entropija mjeri neodređenost sistema? Posmatrajmo sljedeći primjer. Pretpostavimo događaj da kiša pada u Sahari. To se dešava sa jako malom vjerovatnoćom, recimo 0.01 (0.99 je da kiša ne pada). Tada je ovaj događaj sasvim malo neodređen i njegova entropija je $H(X) = 0.0808$ bita. Vjerovatnoća događaja da kiša pada u Britaniji je, recimo, 0.5. To daje maksimalnu neodređenost i entropija ovog sistema je $H(X) = 1$ bit. Dakle, što je entropija (neodređenost) veća, to je veća količina informacije koja se dobije na osnovu opisa događaja vremenskog stanja. Podsjetimo se da mi u principu ne znamo kolika je vjerovatnoća nekog događaja, ali da, na osnovu snimanja velikog broja slučajeva, možemo vršiti procjenu (estimaciju).

Ako imamo binarni slučajni proces (binarni alfabet) koji uzima jednu vrijednost sa vjerovatnoćom p, a drugu sa vjerovatnoćom $1-p$, njegova entropija je jednaka:

$$H(X) = -p\log p - (1-p)\log(1-p).$$

Zbog česte primjene, entropija ovakvog događaja označava se sa $H(p)$ i naziva se **binarnom entropijom**. Sa Slike II.1 jasno je (a kasnije će biti i dokazano) da je entropija najveća za $p = 0.5$ (jednako vjerovatne događaje) kada iznosi $H(p) = 1$ bit.



Slika II.1. Entropija $H(p)$

Posmatrajmo ponovo primjer sa kišom u Sahari. Ako je vjerovatnoća događaja $p = 0.01$, ovaj događaj nosi veliku informaciju kada se dogodi $-\log_2(0.01) = 6.64$; međutim, njegov doprinos entropiji je mali pošto se dešava s malom vjerovatnoćom. Slična situacija je kod glasova u našem jeziku. Samoglasnici se prilično često javljaju sa totalnom vjerovatnoćom koja je približno 0.45 (približno 0.09 je prosječna vjerovatnoća svakog samoglasnika, najveća za 'A'), dok suglasnici imaju ukupnu vjerovatnoću 0.55 (nešto iznad 0.02 u prosjeku, što je znatno manje od samoglasnika; i ovdje se znatno češće pojavljuju neki u odnosu na druge). Dakle, doprinos svakog suglasnika entropiji manji je od doprinosa samoglasnika, ali kada se pojavi suglasnik, on nosi mnogo više informacija od samoglasnika. Posmatrajmo imena „Marko" i „Milijana", gdje su na jednoj strani izdvojeni suglasnici, a na drugoj samoglasnici. Očigledno je lakše pogoditi o kojem je imenu riječ na osnovu grupe suglasnika nego grupe samoglasnika:

Ime	**Suglasnici**	**Samoglasnici**
MARKO	MRK	AO
MILIJANA	MLJN	IIAA

Naredne tri definicije odnose se na entropije vezane za međusobne odnose dva događaja (alfabeta) X i Y.

Definicija II.2: Združena entropija događaja (X,Y), $X = \{x_1, x_2, \ldots, x_N\}$ i $Y = \{y_1, y_2, \ldots, y_M\}$, koji se mogu opisati združenom funkcijom vjerovatnoće $p(x_i,y_j)$, jednaka je:

$$H(X,Y) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log p(x_i, y_j).$$

Definicija II.3: Uslovna entropija $H(Y|X)$ se definiše kao:

$$H(Y \mid X) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log p(y_j \mid x_i).$$

Važi i:

$$H(X \mid Y) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log p(x_i \mid y_j).$$

Definicija II.4: Uslovna entropija $H(Y|X=x)$, pod uslovom da znamo da je ishod događaja iz alfabeta X jednak x, definiše se kao:

$$H(Y \mid X = x) = -\sum_{j=1}^{M} p(y_j \mid x)\log p(y_j \mid x).$$

Uslovna i združena entropija su nenegativne veličine. U oba slučaja imamo u sumama nepozitivni logaritam vjerovatnoće koja pripada intervalu [0,1]. Ovo se množi sa nenegativnom veličinom (vjerovatnoćom). Stoga sumiramo nepozitivne veličine, koje na kraju množimo sa -1, čime sigurno dobijamo nenegativan rezultat.

Tvrdnja II.1: Maksimalna vrijednost uslovne entropije postiže se kada je $H(Y \mid X) = H(Y)$.

Dokaz: Pođimo od:

$$H(Y \mid X) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log p(y_j \mid x_i).$$

Posmatrajmo sada razliku $H(Y|X) - H(Y)$:

$$H(Y \mid X) - H(Y) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log p(y_j \mid x_i) + \sum_{j=1}^{M} p(y_j)\log p(y_j).$$

Koristeći činjenicu (marginalnu vjerovatnoću) da je:

$$\sum_{i=1}^{N} p(x_i, y_j) = p(y_j),$$

drugu sumu u prethodnom izrazu možemo prikazati kao:

$$H(Y \mid X) - H(Y) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log p(y_j \mid x_i) + \sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log p(y_j) =$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log\frac{p(y_j)}{p(y_j \mid x_i)}.$$

Od ove tačke nadalje, dokaz može da ide u raznim smjerovima, ali je najjednostavnije koristiti sljedeću osobinu logaritamske funkcije $\log x \le (x-1)\log e$. Provjera ove nejednakosti je relativno jednostavna jer se može pokazati da funkcija

$$f(x) = \log x - (x-1)\log e$$

dostiže maksimum za $a = 1$ koji iznosi 0. Provjerimo izvod ove funkcije:

$$f'(x) = \frac{1}{x \ln 2} - \log e = \frac{\log e}{x} - \log e = \log e \frac{1-x}{x} = 0.$$

Ovdje je korišćena činjenica da je $\log_a b = 1/\log_b a$. Drugi izvod ove funkcije je:

$$f''(x) = \log e \left(\frac{x-1}{x^2} - \frac{1}{x} \right),$$

te u tački nule prvog izvoda iznosi $f''(1) = -\log e < 0$, što znači da je stvarno predmetna funkcija maksimalna za $x = 1$. Slika II.2 potvrđuje ovu osobinu logaritma. Bitno je ukazati na činjenicu, koja će kasnije biti korišćena, da ova funkcija dostiže maksimum za argument logaritma koji je jednak 1.

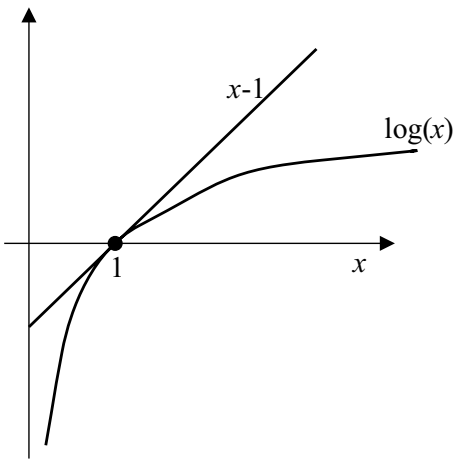
Odavde slijedi:

$$H(Y \mid X) - H(Y) = \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \log \frac{p(y_j)}{p(y_j \mid x_i)} \leq \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \left[\frac{p(y_j)}{p(y_j \mid x_i)} - 1 \right] \log e$$

$$= \log e \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \frac{p(y_j)}{p(y_j \mid x_i)} - \log e \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) =$$

$$= \log e \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i) p(y_j \mid x_i) \frac{p(y_j)}{p(y_j \mid x_i)} - \log e = \log e \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i) p(y_j) - \log e =$$

$$= \log_e \sum_{i=1}^{N} p(x_i) \sum_{j=1}^{M} p(y_j) - \log e = \log e - \log e = 0. \qquad \square$$



Slika II.2. Odnos funkcije $x - 1$ i $\log(x)$ korišćen u dokazu Tvrdnje II.1.

U dokazu su korišćene osobine združene vjerovatnoće, odnosno da se može napisati kao proizvod marginalne i uslovne $p(x_i, y_j) = p(x_i)p(y_j \mid x_i)$, zatim osobina vjerovatnoće i združene vjerovatnoće da je suma po svim mogućim argumentima jednaka 1, te smo konačno razdvojili sume po i i j u pogodnom trenutku, kada argumenti nisu međusobno zavisili od indeksa po kojima je vršeno sumiranje.

Ovim je dokazano da važi

$$H(Y|X) \leq H(Y).$$

Dakle, uslovna entropija je manja od entropije pojedinačnog događaja. Riječ je o logičnoj željenoj osobini entropijskih funkcija, koja je ovdje pokazana. Naime, ako posmatramo neki slučajni događaj Y i saznamo nešto o njemu preko slučajnog događaja X, sigurno umanjujemo neodređenost o događaju Y, odnosno u ovom postupku ne može da dođe do povećanja neodređenosti. Do jednakosti dolazi kada je argument logaritma, na koga smo primjenjivali poređenje, jednak 1, odnosno kada je:

$$p(y_j) = p(y_j \mid x_i).$$

Prethodna relacija važi sa jednakošću samo kada su događaji iz alfabeta X i Y nezavisni! Ponovo je riječ o logičnom zaključku da, kada slučajna promjenljiva X ništa ne govori o slučajnoj promjenljivoj Y, entropija (neodređenost) o slučajnoj promjenljivoj Y ostaje ista, to jest da se ne smanjuje.

Predmetna tvrdnja se u praksi pojavljuje na različite načine: kao entropija uzastopnih karaktera ili simbola u nekoj poruci ili kao odnos između entropije na strani predaje i prijema u komunikacijama. Kod niza simbola ili događaja, ova tvrdnja se može dalje generalizovati. Naime, pretpostavimo da imamo događaje X_1, X_2, ..., X_n. Možemo da tvrdimo:

$$H(X_1) \geq H(X_1 \mid X_2) \geq H(X_1 \mid X_2X_3) \geq \cdots \geq H(X_1 \mid X_2X_3 \cdots X_n).$$

Ova serija nejednakosti često se naziva **lančanim pravilom** (engl. *chain rule*) za entropiju. Može se lako protumačiti da svako novo znanje smanjuje neodređenost o nekom događaju. Kada su događaji međusobno nezavisni, neznanje (entropija) ostaje na istom nivou.

II.1.2 Relativna entropija i međusobna informacija

Definicija II.5: Relativna entropija (ili **Kalbek–Lejblerova** (engl. *Kullback–Leibler*) **distanca** ili **diskriminaciona funkcija** ili **divergencija**) između dvije (diskretne) funkcije $p(x)$ i $q(x)$ definiše se kao (obično se primjenjuje na vjerovatnoće iz diskretnog skupa ili funkcije gustina raspodjele):

$$D(p \,||\, q) = \sum_{i=1}^{N} p(x_i) \log \frac{p(x_i)}{q(x_i)}.$$

Kod ove veličine uvodimo dvije konvencije: da je $0\log(0/q) = 0$ i da $p\log(p/0) = \infty$.

Definicija II.6: Međusobna informacija $I(X;Y)$ između dvije diskretne slučajne promjenljive X i Y, sa združenom gustinom raspodjele $p(x,y)$ i marginalnim raspodjelama $p(x)$ i $p(y)$, definiše se kao:

$$I(X;Y) = \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)} = D(p(x,y) \,||\, p(x)p(y)).$$

Međusobna informacija se može prikazati u funkciji entropija. Ona predstavlja dio informacije koju dijele, odnosno koja je zajednička događajima X i Y.

Lema II.1: Međusobna informacija zadovoljava:

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) = H(X) + H(Y) - H(X,Y).$$

Dokaz:

$$I(X;Y) = \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)} = \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \log \frac{p(x_i)p(y_j \mid x_i)}{p(x_i)p(y_j)} =$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \log \frac{p(y_j \mid x_i)}{p(y_j)} =$$

$$= -\sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \log p(y_j) + \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \log p(y_j \mid x_i).$$

Sada možemo da iskoristimo marginalnu osobinu

$$\sum_{i=1}^{N} p(x_i, y_j) = p(y_j)$$

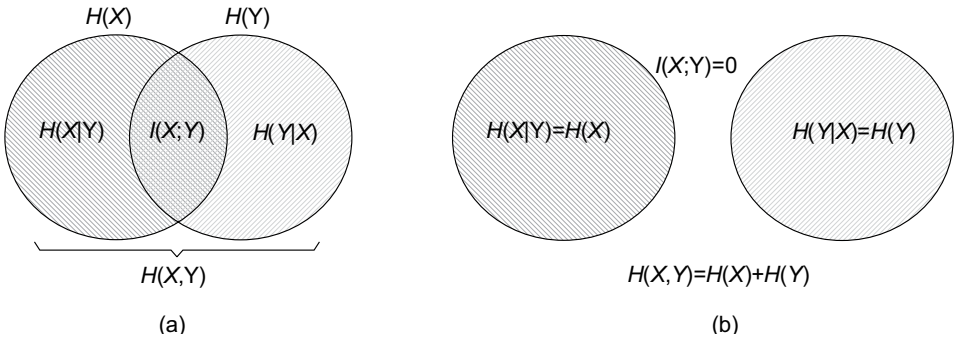
i da je uvrstimo u prvi član prethodnog izraza, čime dobijamo:

$$I(X;Y) = -\sum_{j=1}^{M} p(y_j) \log p(y_j) + \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i, y_j) \log p(y_j \mid x_i) = H(Y) - H(Y \mid X).$$

Na sličan način se mogu dokazati i ostala dva izraza (stava) ove leme.□

Na osnovu ove leme može se definisati i lančano pravilo za entropije:

$$H(X,Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y).$$



Slika II.3. Ilustracija odnosa između entropija i međusobne informacije: (a) Događaji X i Y su zavisni; (b) Događaji X i Y su nezavisni

Ovo pravilo se može formulisati kao: entropija (neodređenost) združenog događaja jednaka je entropiji (neodređenosti) prvog događaja plus neodređenost drugog događaja kada znamo ishod prvog događaja. Jasno je da za nezavisne događaje slijedi:

$$H(X,Y) = H(X) + H(Y).$$

Lančano pravilo za entropije može se generalizovati i za veći broj događaja:

$$H(X_1, X_2, ..., X_n) = H(X_1) + H(X_2 \mid X_1) + H(X_3 \mid X_1 X_2) + \cdots + H(X_n \mid X_1 X_2 \cdots X_{n-1}).$$

Međusobna informacija, kao i relativna entropija, nenegativne su veličine:

$$D(p \,||\, q) \geq 0$$

$$I(X;Y) \geq 0.$$

Dokaz prve tvrdnje nije trivijalan kao u slučaju entropija. Dokaz druge tvrdnje se može obaviti posredno na osnovu Leme II.1:

$$I(X;Y) = H(X) - H(X \mid Y).$$

Kako je $H(X) \geq H(X \mid Y)$, odavde slijedi da je $I(X;Y) = H(X) - H(X \mid Y) \geq 0$.

Na Slici II.3 prikazana je ilustracija u obliku Venovih dijagrama, koja se može upotrijebiti za pamćenje relacija među entropijama i međusobnom informacijom. Združena entropija je jednaka ukupnoj površini koju zapremaju osjenčene površi, dok je međusobna informacija u presjeku. Za slučaj sa slike (b) predmetni „skupovi“ nemaju presjek, pa je združena entropija jednaka sumi entropija $H(X) + H(Y)$, a pošto nema presjeka, međusobna informacija je $I(X;Y) = 0$.

Primjer II.1. Posmatrajmo slučaj bacanja kocke iz Primjera I.2. Odrediti entropije događaja X i Y, uslovne entropije, združenu entropiju i međusobnu informaciju.

Rješenje: Entropiju združenog događaja lako određujemo na osnovu Tabele I.2 s vjerovatnoćama združenog događaja $p(x,y)$, uočavajući da imamo 21 nenultu vjerovatnoću, od kojih je šest vjerovatnoća 1/36 i 15 vjerovatnoća 2/36.

$$H(X,Y) = -6 \cdot \frac{1}{36}\log_2\frac{1}{36} - 15 \cdot \frac{2}{36}\log_2\frac{2}{36} \approx 4.33 \text{ bita.}$$

Iz iste tabele, korišćenjem marginalnih vjerovatnoća, dobijamo entropije pojedinačnih događaja:

$$H(X) = -2 \cdot \frac{1}{36}\log_2\frac{1}{36} - 2 \cdot \frac{2}{36}\log_2\frac{2}{36} - 2 \cdot \frac{3}{36}\log_2\frac{3}{36} -$$
$$- 2 \cdot \frac{4}{36}\log_2\frac{4}{36} - 2 \cdot \frac{5}{36}\log_2\frac{5}{36} - \frac{6}{36}\log_2\frac{6}{36} \approx 3.27 \text{ bita}$$

$$H(Y) = -\frac{1}{36}\log_2\frac{1}{36} - \frac{3}{36}\log_2\frac{3}{36} - \frac{5}{36}\log_2\frac{5}{36} - \frac{9}{36}\log_2\frac{9}{36} - \frac{11}{36}\log_2\frac{11}{36} \approx 2.32 \text{ bita}.$$

Entropije uslovnih događaja možemo sračunati korišćenjem definicione formule i tabela združene i uslovne vjerovatnoće. Međutim, sada je jednostavnije da koristimo relacije:

$$H(X,Y) = H(X) + H(Y \mid X) \Rightarrow H(Y \mid X) = H(X,Y) - H(X) \approx 1.06 \text{ bita}$$
$$H(X \mid Y) = H(X,Y) - H(Y) \approx 2.01 \text{ bita.}$$

Konačno, međusobna informacija je jednaka:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = 3.27 + 2.32 - 4.33 = 1.26 \text{ bita.} \quad \square$$

Vidjeli smo da je kod binarnog događaja entropija maksimizovana za slučaj kada su simboli alfabeta jednako vjerovatni. Postavlja se pitanje: koje su granice entropije kod opšteg N-arnog alfabeta X? Očigledno je da je minimalna entropija jednaka nuli i da se ona postiže kada imamo jedan siguran i ostale nemoguće događaje, odnosno kada nema neodređenosti.

Teorema II.1. Maksimum entropije N-arnog alfabeta postiže se za jednakovjerovatne simbole (uniformnu raspodjelu) i iznosi $\log N$.

Dokaz: Teorema će biti dokazana na dva načina. Prvi način je duži i počiva na osobini nenegativnosti relativne entropije, koju nismo do sada pokazali. Posmatrajmo entropiju:

$$H(X) = -\sum_{i=1}^{N} p_i \log p_i = H(p_1, p_2, ..., p_N).$$

Izrazili smo je kao sumu N promjenljivih p_i koje predstavljaju vjerovatnoće pojavljivanja pojedinih simbola. Pronalaženje ekstremne tačke (minimuma i maksimuma)

ovakve funkcije podrazumijeva određivanje parcijalnih izvoda po promjenljivim i izjednačavanje s nulom. Međutim, direktni postupak će nas dovesti u ćorsokak:

$$\frac{\partial H(p_1,p_2,....,p_N)}{\partial p_j} = -\log p_j - \log e = 0 \Rightarrow p_j = 1/e \quad \forall j \in [1,N].$$

Dobili smo (što će se ispostaviti kao tačno) da su sve vjerovatnoće jednake, ali i paradoks da je suma vjerovatnoća različita od jedan:

$$\sum_{i=1}^{N} p_i = \frac{N}{e} \neq 1.$$

Razlog za pojavu ovog paradoksa leži u činjenici da nisu sve vjerovatnoće međusobno nezavisne, već da između njih postoji veza (ograničenje):

$$\sum_{i=1}^{N} p_i = 1 \Rightarrow p_N = 1 - p_1 - p_2 - \cdots - p_{N-1}.$$

Dakle, entropija nije funkcija N-nezavisnih promljenljivih, već $N-1$ nezavisne i jedne (uzeli smo N-te) zavisne. Optimizacija ovakve funkcije može se izvesti na dva načina. Pošto se oba intenzivno koriste u ovoj oblasti, dajemo ih oba.

I način. Prikažimo entropiju kao funkciju $N-1$ nezavisne promjenljive:

$$H(p_1,p_2,...,p_{N-1}) = -\sum_{i=1}^{N-1} p_i \log p_i - (1-p_1-p_2-\cdots-p_{N-1})\log(1-p_1-p_2-\cdots-p_{N-1}).$$

Sada odredimo parcijalne izvode i izjednačimo ih s nulom:

$$\frac{\partial H(p_1,p_2,....,p_{N-1})}{\partial p_j} = -\log p_j - \log e + \log(1-p_1-p_2-\cdots-p_{N-1}) + \log e = 0 \Rightarrow$$

$$p_j = 1 - \sum_{i=1}^{N-1} p_i, \quad \forall j \in [1,N-1].$$

Dakle, dobili smo ponovo da je svako p_j za $j \in [1,N-1]$ međusobno jednako i da je jednako $1-\sum_{i=1}^{N-1} p_i = 1-(N-1)p_j$, pa na osnovu ovoga slijedi da je:

$$p_j = 1 - \sum_{i=1}^{N-1} p_i = 1-(N-1)p_j \Rightarrow Np_j = 1 \Rightarrow p_j = \frac{1}{N} \quad \forall j \in [1,N-1].$$

Stoga je lako uočiti da je maksimalna entropija jednaka:

$$H(X) = H\left(\frac{1}{N},\frac{1}{N},...,\frac{1}{N}\right) = -(N-1)\sum_{i=1}^{N-1}\frac{1}{N}\log\frac{1}{N} - \frac{1}{N}\log\frac{1}{N} =$$

$$= -\frac{N}{N}\log\frac{1}{N} = \log N.$$

II način. Dodavanjem na entropijsku funkciju vrijednosti koja je jednaka nula, njen iznos se ne mijenja. U našem slučaju znamo da je suma vjerovatnoća jednaka 1, pa dodavanjem člana

$$\lambda\left(\sum_{i=1}^{N} p_i - 1\right),$$

gdje se λ naziva Lagranžeovim (fr. *Lagrange* – plemić, matematičar, filozof i fizičar s kraja XVIII i početka XIX vijeka) množiocem (predmetni postupak je uobičajena tehnika optimizacije s ograničenjem, a potreba za uvođenjem ovakvog množioca biće jasna nešto kasnije). Funkcija koju optimizujemo može se sada zapisati kao:

$$h(p_1, p_2, ..., p_N) = H(p_1, p_2, ..., p_N) + \lambda\left(\sum_{i=1}^{N} p_i - 1\right) =$$

$$= -\sum_{i=1}^{N} p_i \log p_i + \lambda\left(\sum_{i=1}^{N} p_i - 1\right).$$

Odredimo sada parcijalne izvode i izjednačimo ih s nulom:

$$\frac{\partial h(p_1, p_2, ..., p_N)}{\partial p_j} = -\log p_j - \log e + \lambda = 0 \Rightarrow p_j = \frac{2^{\lambda}}{e} \quad \forall j \in [1, N].$$

Sada je (ponovo) jasno da se odgovarajući optimum dobija kada su sve vjerovatnoće jednake, a Lagranžeov množilac je, da bi rezultat bio moguć, odnosno da bi bilo zadovoljeno odgovarajuće ograničenje:

$$\frac{2^{\lambda}}{e} = \frac{1}{N} \Rightarrow 2^{\lambda} = \frac{e}{N} \Rightarrow \lambda = \log_2\left(\frac{e}{N}\right).$$

Uvrštavanjem dobijamo ponovo isti rezultat kao gore − da je maksimalna entropija jednaka log*N*. Poznato je da nule prvog izvoda mogu da korespondiraju minimumu, maksimumu i prevojnoj tački. Radi matematičke rigoroznosti možemo provjeriti što se dešava u našem slučaju, a ta se provjera može obaviti putem drugih (parcijalnih) izvoda:

$$\frac{\partial^2 h(p_1, p_2, ..., p_N)}{\partial p_j^2} = -\frac{1}{p_j}\log e < 0 \quad \text{za } p_j = \frac{1}{N}.$$

Dakle, pošto je drugi izvod manji od nule, jasno je da se radi o maksimumu odgovarajuće funkcije. Da sublimiramo, entropija se nalazi u granicama:

$$0 \le H(X) \le \log N,$$

gdje se donja granica dobija kada u alfabetu imamo siguran događaj, dok se gornja granica dobija za jednakovjerovatne ishode. Uvijek moramo imati na umu uslovno smanjivanje entropije kao još jednu njenu bitnu karakteristiku:

$$H(X \mid Y) \leq H(X). \ \square$$

II.1.3 Markovljev lanac i Fanoova nejednakost*

Definicija II.7: Slučajne promjenljive formiraju (**Markovljev**) **lanac**, označen sa $X \rightarrow Y \rightarrow Z$ ako:

$$p(z|x,y) = p(z|y),$$

to jest Z je uslovno nezavisna od X za dato Y, ili preciznije rečeno, ako imamo informaciju vezanu za Y, informacija o X nam ne daje ništa novo o slučajnoj promjenljivoj Z. Dalje, $X \rightarrow Y \rightarrow Z$ znači: $p(x,y|,z) = p(x)p(y|x)p(z|y)$. □

Pretpostavimo da smo poslali informaciju X prema prijemniku koji prima informaciju Y zbog izobličenja u komunukacionom kanalu. Zatim prosljeđuje ovu informaciju dalje, ka narednom učesniku u komunikaciji koji prima informaciju Z. Za ovakav sistem važi Teorema II.2, koja se naziva nejednakošću u procesiranju informacija (engl. *data processing inequality*).

Posljedica II.1: $X \rightarrow Y \rightarrow Z$ akko su X i Z uslovno nezavisni za dato Y. Ovo slijedi iz:

$$p(x,z \mid y) = \frac{p(x,y,z)}{p(y)} = \frac{p(x,y)p(z \mid x,y)}{p(z)} = \frac{p(x,y)p(z \mid y)}{p(z)} = p(x \mid y)p(z \mid y).$$

Očigledno $X \rightarrow Y \rightarrow Z$ implicira $Z \rightarrow Y \rightarrow X$.

Teorema II.2. Nejednakost u procesiranju informacija. U komunikaciji $X \rightarrow Y \rightarrow Z$ važi $I(X;Y) \geq I(X;Z)$.

Drugim riječima, ova teorema kaže da ako je Z nastalo procesiranjem podataka nad Y, $Z = F(Y)$, bilo da je $F()$ deterministička ili slučajna funkcija nezavisna od X, ne dolazi do uvećanja znanja koje nam Y može saopštiti o X.

Dokaz: Međusobna informacija može se zapisati kao:

$$I(X;Y,Z) = I(X;Z) + I(X;Y|Z) = I(X;Y) + I(X;Z|Y).$$

Kako su X i Z nezavisni za dato Y, tada slijedi da je: $I(X;Z|Y) = 0$. Kako je $I(X;Y|Z) \geq 0$, tada važi:

$$I(X;Y) \geq I(X;Z). \quad \square$$

Ova nejednakost ima značaj koji prevazilazi teoriju informacija. Na primjer, u telekomunikacijama mi vršimo filtriranje korisnog signala zato što znamo da se

on nalazi na nekoj frekvenciji ili kanalu, uklanjajući šum koji se ne nalazi na predmetnoj frekvenciji. Međutim, ako ne znamo gdje se korisni signal nalazi, niti imamo način da odredimo njegovu poziciju, najbolje je ne raditi ništa, odnosno ne vršiti nikakvu operaciju bez obezbijeđenog predznanja. U mnogim oblastima inženjerskih nauka često se pravi greška procesiranjem signala o kome ne postoji predznanje (ili postoji u nedovoljnom obimu), čime se informacije pohranjene u signalu mogu samo izgubiti. Dakle, poruka ove bitne nejednakosti je da je jedino ispravno ono procesiranje koje može da obezbijedi popravku kvaliteta primljenog signala ili mjerenja kada imamo predznanje o pojavi koja se opservira ili kada smo takvo saznanje stekli.

Ovim smo priveli kraju osnovne pojmove vezane za entropiju i međusobnu informaciju. Jedan dio jednako bitnih elemenata ostavljen je za razradu, u okviru zadataka za vježbu na kraju poglavlja. Vrijedi napomenuti da je moguće kombinovanje većeg broja slučajnih promjenljivih direktno i kao uslovno zavisnih. Na primjer, moguće je na sljedeći način definisati međusobnu informaciju dva događaja X i Y, pod uslovom da se dogodio treći događaj Z:

$$I(X;Y \mid Z) = H(X \mid Z) - H(X \mid Y,Z) = H(Y \mid Z) - H(Y \mid X,Z) = \\ = H(X \mid Z) + H(Y \mid Z) - H(X,Y \mid Z)$$

gdje je, na primjer, $H(X \mid Y,Z)$ uslovna entropija događaja X, pod uslovom da su se dogodili Y i Z. Slične generalizacije lančanog pravila moguće su i za druge kombinacije slučajnih događaja (promjenljivih).

Fundamentalni problem u nauci, a posebno u komunikacijama, jeste procjena (estimacija) neke vrijednosti na osnovu odgovarajućeg mjerenja. Pretpostavimo da je vrijednost X poslata u komunikacioni kanal, gdje je promijenjena, i da je primljena vrijednost Y. Naša je želja da na osnovu primljene vrijednosti procijenimo vrijednost X. Procijenjene vrijednosti ćemo označavati kao $\hat{X}$. Dakle, procjena se može opisati sljedećom funkcijom: $\hat{X} = g(Y)$.

Naravno, postoji potreba da se odredi koja je vjerovatnoća da je naša procjena jednaka pravoj vrijednosti. Jedan način da se ovo opiše je poznata **Fanoova nejednakost**, koja povezuje vjerovatnoću greške s uslovnom entropijom $H(X|Y)$. Intuitivno govoreći, ako je mala neodređenost o vrijednosti X kada imamo Y, onda je vjerovatnoća greške mala i obrnuto. Entropija $H(X|Y) = 0$ kaže da bi vjerovatnoća greške u tom slučaju trebala biti 0, odnosno, nakon opservacije Y mi nemamo neodređenosti oko vrijednosti X. Fanoova nejednakost vrši kvantifikovanje ovih tvrdnji. Uvedimo, prije teoreme, sljedeću oznaku:

$$P_e = \Pr\{\hat{X} \neq X\}.$$

Teorema II.3. (Fanoova nejednakost):

$$H(P_e) + P_e \log(||A|| - 1) \geq H(X|Y).$$

Teorema se može redukovati na: $1 + P_e \log(||A||) \geq H(X|Y)$, gdje je $||A||$ kardinalnost posmatranog alfabeta.

Napomena: Za $P_e = 0$ ova se teorema svodi na $H(X|Y) = 0$.

Dokaz: Definišimo slučajnu promjenljivu E kao:

$$E = \begin{cases} 1 & X \neq \hat{X} \\ 0 & X = \hat{X} \end{cases}.$$

Sada se $H(E,X|Y)$ može razviti na dva načina, korišćenjem lančanog pravila:

$$H(E,X|Y) = H(X|Y) + H(E|X,Y) = H(E|Y) + H(X|E,Y).$$

Dalje, $H(E|X,Y) = 0$, jer ako je poznato i X i Y, mi ne pravimo grešku. Takođe, na osnovu pravila o redukovanju entropije, slijedi: $H(E|Y) \leq H(E) = H(P_e)$.

$$H(X|E,Y) = \Pr(E=0)H(X|Y,E=0) + \Pr(E=1)H(X|Y,E=1) \leq (1-P_e)0 + \\ + P_e \log(||A|| - 1).$$

Odavde slijedi:

$$H(X|Y) + H(E|X,Y) = H(E|Y) + H(X|E,Y) \leq H(P_e) + P_e \log(||A|| - 1),$$

čime je dokaz obavljen. □

Fanoova nejednakost se može tumačiti na sljedeći način. Prilikom prijema na izlazu potrebno je $H(P_e)$ da bi se provjerilo da li je došlo do greške, kao i još $\log(||A|| - 1)$ da bi se, u slučaju da je do greške došlo, provjerilo koji je simbol zapravo primljen.

II.2 Markovljevi procesi

II.2.1 Značaj Markovljevih procesa

Glas (slovo) koji trenutno izgovaramo zavisi od prethodnog glasa. Nakon glasa ‘c’ mala je šansa da se, na primjer, pojavi neki suglasnik, a mnogo veća da bude samoglasnik, a ako imamo na umu, na primjer, suglasnike, onda je izbor ponovo dosta sužen. Dalje, ‘a’ je najčešće slovo u našem jeziku, ali se dosta rjeđe pojavljuje iza samoglasnika nego što je njegovo pojavljivanje u nekom tekstu (knjizi, pismu, govoru itd.). Ako je neko izgovorio „zak“, razumnih nastavaka ima samo nekoliko, npr.: ‘a’, ‘o’, ‘lj’ itd., a mnogo je više glasova koji ne mogu nastaviti ovako započetu riječ. Dakle, vjerovatnoća pojavljivanja nekog simbola u alfabetu ima u teoriji informacija limitirani značaj, dok je jednaka, ako ne i mnogo veća,

važnost uslovnih vjerovatnoća među, na primjer, uzastopnim simbolima jednog alfabeta. Kao što ćemo vidjeti kasnije, ovakve uslovne vjerovatnoće (korelacije) nam omogućavaju smanjivanje potrebnog memorijskog prostora za smještaj informacija. Naravno, ove pogodnosti korišćenja uslovnih vjerovatnoća plaćamo povećanjem memorijskih i računskih zahtjeva sistema.

II.2.2 Definicija

Definicija II.8: Markovljevim sistemom nultog reda podrazumijevamo sistem koji je u potpunosti determinisan vjerovatnoćama simbola datog alfabeta i gdje ne postoji uslovljenost pojave ovih simbola susjednim simbolima $p(x_i)$, $i \in [1,N]$. Ovakvi sistemi (procesi) nazivaju se i **sistemima (procesima) bez memorije**. **Markovljevi sistemi prvog reda** podrazumijevaju sisteme koji su determinisani uslovnim vjerovatnoćama prvog reda $p(s_j \mid s_{j-1})$, gdje je $s_j, s_{j-1} \in \{x_i \mid i \in [1,N]\}$. **Markovljevi sistemi *k*-tog reda** su determinisani uslovnim vjerovatnoćama *k*-tog reda $p(s_j \mid s_{j-1}, s_{j-2}, ..., s_{j-k})$, gdje je $s_j, s_{j-1}, ..., s_{j-k} \in \{x_i \mid i \in [1,N]\}$.

II.2.3 Način prikaza

Kod Markovljevih procesa podrazumijeva se (što je i prirodno) da vjerovatnoća pojavljivanja nekog simbola u velikoj mjeri zavisi od prethodnih (ili budućih) stanja. Kao takvi, Markovljevi procesi su pogodni za modelovanje mnogih realističnih pojava, uključujući one od interesa za teoriju informacija i kodova. Za karakterizaciju Markovljevog procesa nultog reda potrebno je N vjerovatnoća, za procese prvog reda N^2 uslovnih vjerovatnoća, dok je za procese *k*-tog reda potrebno znati N^{k+1} uslovnih vjerovatnoća *k*-tog reda.

Postoje tri načina da se prikažu Markovljevi procesi:

(a) matricom tranzicije,

(b) grafom tranzicije i

(c) trelisom.

U različitim situacijama i za razne sisteme jedan od ovih načina je pogodniji nego ostali. Pretpostavimo da su u nekom posmatranom trenutku vjerovatnoće pojave nekog simbola opisane putem vektora vjerovatnoća:

$$\mathbf{p} = [p_1, p_2, ..., p_N] = [P(x_1), P(x_2), ..., P(x_N)].$$

Markovljev sistem prvog reda može se opisati uslovnim vjerovatnoćama $P(s_j = x_i \mid s_{j-1} = x_k)$, $i \in [1,N]$, $k \in [1,N]$, a ovo je zgodno prikazati **matricom tranzicije**, to jest matricom uslovnih vjerovatnoća:

$$\mathbf{P}=\begin{bmatrix} P(x_1\mid x_1) & P(x_2\mid x_1) & \cdots & P(x_N\mid x_1)\\ P(x_1\mid x_2) & P(x_2\mid x_2) & \cdots & P(x_N\mid x_2)\\ \vdots & \vdots & \ddots & \vdots\\ P(x_1\mid x_N) & P(x_2\mid x_N) & \cdots & P(x_N\mid x_N)\end{bmatrix}.$$

Dakle, kolone (ovdje smo usvojili takvu konvenciju, ali može i drugačije da se radi) predstavljaju prethodne simbole, dok vrste predstavljaju tekuće simbole. Kako za svaki prethodni simbol moramo dobiti neki simbol na ulazu, to znači da je suma vrsta tranzicione matrice jednaka jedan (sigurnom događaju):

$$\sum_{j=1}^{N} P(x_j\mid x_i)=1 \text{ za svako } i\in[1,N].$$

Osnovna mana matrice tranzicije je činjenica da se teško primjenjuje kod Markovljevih sistema višeg reda. Na primjer, ako imamo Markovljev sistem drugog reda, uslovne vjerovatnoće u matrici tranzicije su $P(s_j=x_i\mid s_{j-1}=x_k, s_{j-2}=x_l)$, $i\in[1,N]$, $k\in[1,N], l\in[1,N]$, što se u obliku matrice može prikazati kao:

$$\mathbf{P}=\begin{bmatrix} P(x_1\mid x_1x_1) & P(x_2\mid x_1x_1) & \cdots & P(x_N\mid x_1x_1)\\ P(x_1\mid x_1x_2) & P(x_2\mid x_1x_2) & \cdots & P(x_N\mid x_1x_2)\\ \vdots & \vdots & \ddots & \vdots\\ P(x_1\mid x_1x_N) & P(x_2\mid x_1x_N) & \cdots & P(x_N\mid x_1x_N)\\ P(x_1\mid x_2x_1) & P(x_2\mid x_2x_1) & \cdots & P(x_N\mid x_2x_1)\\ \vdots & \vdots & \ddots & \vdots\\ P(x_1\mid x_Nx_N) & P(x_2\mid x_Nx_N) & \cdots & P(x_N\mid x_Nx_N)\end{bmatrix}.$$

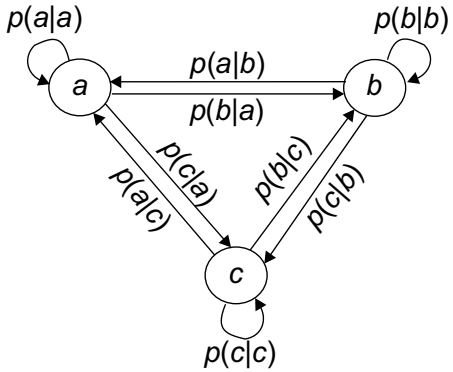
U predmetnom slučaju, matrica tranzicije dimenzija je $N^2\times N$, dok je za Markovljev sistem K-tog reda dimenzija matrice tranzicije bila $N^K\times N$.

Bolja vizuelizacija Markovljevih sistema može se postići putem **grafa tranzicije**. U čvorovima ovoga grafa nalaze se prethodna stanja, dok su (usmjerene) ivice grafa uslovne vjerovatnoće za prelazak iz jednog u drugo stanje.

Primjer II.2. Posmatrajmo alfabet sa tri simbola $X=\{a, b, c\}$. Markovljev sistem I reda, konstruisan nad ovim alfabetom, ima sljedeće uslovne vjerovatnoće (prvog reda):

$$P(a|a)=P(b|a)=P(c|a)=1/3$$
$$P(b|b)=P(c|c)=1/2$$
$$P(a|b)=P(a|c)=P(b|c)=P(c|b)=1/4.$$

Prikazati matricu tranzicije i graf tranzicije za ovaj sistem.



Slika II.4. Graf tranzicije za Markovljev sistema I reda kod ternarnog alfabeta (alfabeta sa tri stanja − simbola)

Rješenje: Matrica tranzicije u ovom slučaju je:

$$\mathbf{P}=\begin{bmatrix} P(a\,|\,a) & P(b\,|\,a) & P(c\,|\,a) \\ P(a\,|\,b) & P(b\,|\,b) & P(c\,|\,b) \\ P(a\,|\,c) & P(b\,|\,c) & P(c\,|\,c) \end{bmatrix}=\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/4 & 1/2 \end{bmatrix}.$$

Graf tranzicije je prikazan na Slici II.4. □

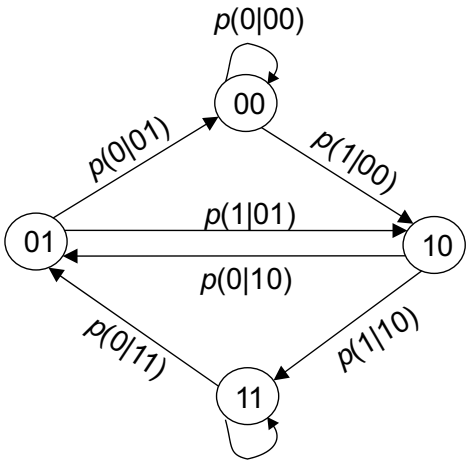
Graf tranzicije ima sličan nedostatak kao i matrica tranzicije, odnosno zbog rasta dimenzija problema teško se primjenjuje kod Markovljevih sistema višeg reda.

Primjer II.3. Posmatrajmo Markovljev sistem II reda sa binarnim alfabetom, s uslovnim vjerovatnoćama $P(0|00)=P(1|11)=0.7$, $P(1|00)=P(0|11)=0.3$, $P(0|01)=P(0|10)=P(1|01)=P(1|10)=0.5$. Prikazati matricu i graf tranzicije za dati sistem.

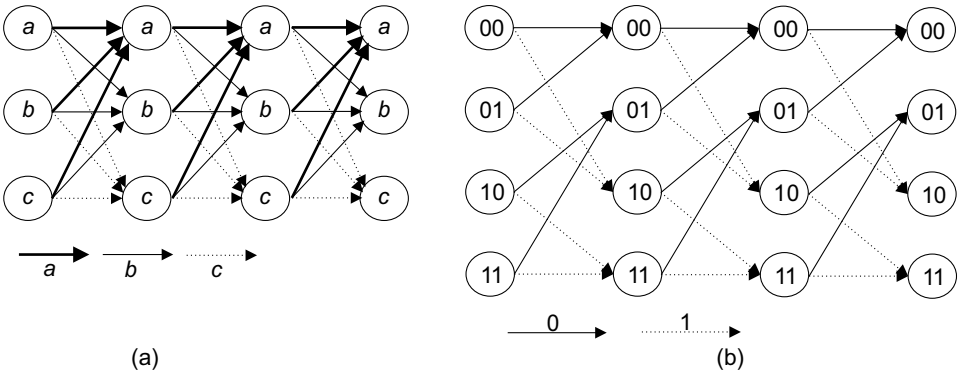
Rješenje: Matrica tranzicije u ovom slučaju je:

$$\mathbf{P}=\begin{bmatrix} P(0\,|\,00) & P(1\,|\,00) \\ P(0\,|\,01) & P(1\,|\,01) \\ P(0\,|\,10) & P(1\,|\,10) \\ P(0\,|\,11) & P(1\,|\,11) \end{bmatrix}=\begin{bmatrix} 0.7 & 0.3 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.3 & 0.7 \end{bmatrix}.$$

Na Slici II.5 prikazan je predmetni graf tranzicije. Uočite da u ovom grafu ne postoje putanje između svih stanja. Naime, ako smo bili u stanju 10 (prethodni simbol da je 1, a prije njega da je 0), možemo preći samo u stanje 11 (dolaskom jedinice) i stanje 01 (dolaskom 0). Napomenimo da se konvencija u smislu redosljeda bita u stanju ponekad mijenja, tako da u nekim udžbenicima možete naći drugačije oznake, npr.: da se prethodni simbol nalazi u stanju kao posljednji u listi, a oni



Slika II.5. Graf tranzicije za Markovljev sistem II reda kod binarnog alfabeta



Slika II.6. Primjeri trelis dijagrama: (a) Ternarni Markovljev sistem prvog reda; (b) Binarni Markovljev sistem drugog reda

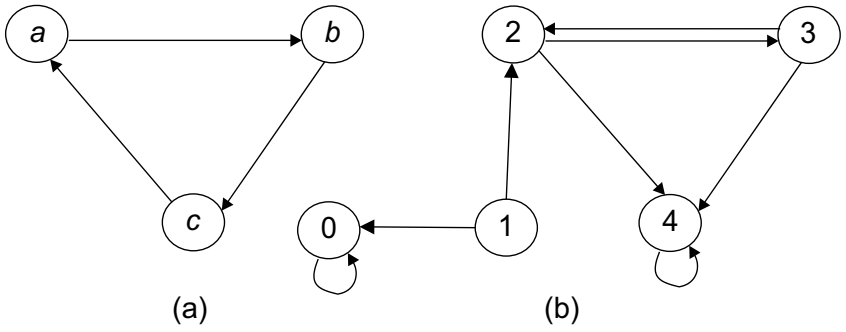
koji su mu prethodili nalaze se prije. U svakom slučaju, to ne mijenja suštinu, ali čitalac mora proučiti konvencije korišćene u materijalu koji koristi.

Da bi se lakše vizuelizovali Markovljevi sistemi višeg reda, uveden je posebni način vizuelizacije grafa koji se naziva **trelis**. Pored ovoga, trelis se koristi i u teoriji informacija, ali i u teoriji kodova da bismo naglasili vremensku dinamiku dešavanja u procesu kodiranja. Na Slici II.6 prikazan je trelis oba uvedena Markovljeva sistema: prvog reda za ternarni alfabet (Slika II.4) i drugog reda za binarni alfabet (Slika II.5). Trelis je često pregledniji jer se može uočiti kako se u vremenskim trenucima odvija kodiranje. U ovom slučaju sve latice su iste, odnosno ponavljaju se iz trenutka u trenutak, ali, kako ćemo vidjeti u Poglavlju VII, to ne mora da bude uvijek tako, pa je vremenska dimenzija često još značajnija. Kod trelisa smo

i naglasili različitim oblicima linija koji simbol izaziva prelazak u novo stanje (kod sistema prvog reda situacija je jednostavna: to je ono stanje u koje se prelazi, dok kod sistema drugog reda to nije toliko očigledno, a, kako ćemo vidjeti kasnije u Poglavlju VII, ovo može biti od velikog značaja).

*II.2.4 Tipovi stanja i sistema**

Markovljev sistem se naziva **nesvodivi** (ponekad se kaže **ergodičan**) ako je moguće da se iz bilo kog stanja pređe u bilo koje stanje u određenom (konačnom) broju koraka. Za dva stanja se kaže da **komuniciraju** među sobom ako se iz prvog može preći u drugo stanje i iz drugog stanja u prvo (u određenom broju koraka). **Komunikacionom klasom** se naziva podskup Markovljevog sistema u kojem svi čvorovi međusobno komuniciraju. Ponekad se ovaj termin koristi samo za najveći podskup Markovljevog sistema u kojem je obezbijeđeno međusobno komuniciranje svih čvorova. Komunikaciona klasa se naziva **zatvorenom** ako se ne može napustiti, odnosno ako ne postoji nikakva mogućnost (vjerovatnoća) da se iz čvorova koji se nalaze unutar klase izađe do čvorova koji se nalaze van klase. Stanje je **rekurentno (esencijalno)** ako se iz svakog stanja, u koje se može doći iz tog stanja, može vratiti u dato stanje. Dakle, u zatvorenoj komunikacionoj klasi sva stanja su esencijalna. Ergodični Markovljevi sistemi predstavljaju jednu komunikacionu klasu. Možemo zaključiti i da su neergodični Markovljevi procesi oni kod kojih postoji makar jedno stanje iz kojeg se ne može preći u neko drugo stanje. Stanje je **periodično** ako postoji cio broj p ($p > 1$), takav da se u isto stanje može vratiti samo poslije kp koraka (k pozitivan cio broj). Napominjemo da problem periodičnosti može biti i komplikovan. Na primjer, postoji mogućnost da se u stanje možemo vratiti samo u koracima 6, 8, 10, 12... Ako je $p = 1$, u pitanju je **aperiodično** stanje, dok ako su sva stanja aperiodična i sam Markovljev sistem je aperiodičan (nema perioda). Stanje se naziva **tranzijentnim** ako postoji nenulta mogućnost da se nakon njegovog napuštanja više ne vratimo u isto stanje. Suprotno



Slika II.7. Primjeri dva Markovljeva sistema: (a) Sistem sa ciklusom sa periodom od tri stanja; (b) Markovljev sistem sa dva apsorbujuća stanja (0 i 4)

od tranzijentnog stanja je **rekurentno** (alternativno se naziva i **perzistentnim**). Završimo ovaj pregled pojmom **apsorbujućeg** stanja koje se ne može napustiti. Napomenimo takođe da postoji mnogo više detalja vezanih za stanja i prikaz Markovljevih sistema koja prevazilaze obim ove knjige (pogledati referencu [2]). Na Slici II.7 prikazan je ciklus od tri stanja, kao i Markovljev sistem sa pet stanja, na kojima se izdvajaju apsorbujuća stanja 0 i 4.

II.2.5 Stacionarno stanje Markovljevog sistema

Ergodični Markovljev sistem je u potpunosti opisan s uslovnim vjerovatnoćama. To znači da je moguće odrediti i sve druge vjerovatnoće, uključujući i vjerovatnoće svakog pojedinačnog stanja. Ovakve vjerovatnoće nazivamo vjerovatnoćama u stacionarnom (stabilnom) stanju iz razloga koji će uskoro biti jasni.

Posmatrajmo prvo Markovljev sistem prvog reda, prikazan putem grafa tranzicije na Slici II.4. Pretpostavimo da u nekom početnom stanju (stanju 0) imamo vjerovatnoće pojedinih simbola alfabeta, date kao $[p^{(0)}(a), p^{(0)}(b), p^{(0)}(c)]$. Želimo da odredimo vjerovatnoće stanja u narednim koracima (nakon što se tekuće stanje napusti i pređe u naredno) $[p^{(i)}(a), p^{(i)}(b), p^{(i)}(c)]$, $i = 1,2\ldots$ Cilj je da se odredi, ako je moguće, vjerovatnoća simbola alfabeta u stacionarnom stanju kada se vjerovatnoća pojedinih simbola više ne mijenja. U sljedeća dva primjera odredićemo stacionarne vjerovatnoće za sistem sa Slike II.4, kao i za sistem drugog reda, konstruisan nad binarnim alfabetom (Slika II.5).

Primjer II.4. Odrediti vjerovatnoće u stacionarnom stanju za Markovljev sistem prvog reda sa ternarnim alfabetom. Odrediti vjerovatnoće simbola u stacionarnom režimu, pod pretpostavkom da je to moguće (a vidjećemo da je, kod ovog ergodičnog sistema, moguće određivanje stacionarnih vjerovatnoća).

Rješenje: Pretpostavimo da su u početnom trenutku $[p^{(0)}(a), p^{(0)}(b), p^{(0)}(c)]$, pa pretpostavimo da želimo da odredimo vjerovatnoće u narednom koraku $[p^{(1)}(a), p^{(1)}(b), p^{(1)}(c)]$:

$$p^{(1)}(a) = P(a \mid a)p^{(0)}(a) + P(a \mid b)p^{(0)}(b) + P(a \mid c)p^{(0)}(c).$$

Dakle, ovim smo sumirali vjerovatnoće da smo se nalazili u nekom od stanja i prešli u stanje a. Na isti način možemo zapisati za ostala dva stanja:

$$p^{(1)}(b) = P(b \mid a)p^{(0)}(a) + P(b \mid b)p^{(0)}(b) + P(b \mid c)p^{(0)}(c)$$
$$p^{(1)}(c) = P(c \mid a)p^{(0)}(a) + P(c \mid b)p^{(0)}(b) + P(c \mid c)p^{(0)}(c),$$

sublimirajući ove tri relacije u matričnom obliku:

$$\begin{bmatrix} p^{(1)}(a) \\ p^{(1)}(b) \\ p^{(1)}(c) \end{bmatrix} = \begin{bmatrix} P(a\,|\,a) & P(a\,|\,b) & P(a\,|\,c) \\ P(b\,|\,a) & P(b\,|\,b) & P(b\,|\,c) \\ P(c\,|\,a) & P(c\,|\,b) & P(c\,|\,c) \end{bmatrix} \begin{bmatrix} p^{(0)}(a) \\ p^{(0)}(b) \\ p^{(0)}(c) \end{bmatrix} \qquad \mathbf{p}^{(1)} = \mathbf{P}^T\mathbf{p}^{(0)},$$

gdje su vektori vjerovatnoća simbola dati kao: $\mathbf{p}^{(i)} = [p^{(i)}(a), p^{(i)}(b), p^{(i)}(c)]^T$. Sada se vjerovatnoća simbola ternarnog alfabeta, u narednim koracima, može kompaktno zapisati kao:

$$\mathbf{p}^{(2)} = \mathbf{P}^T\mathbf{p}^{(1)} = \left[\mathbf{P}^T\right]^2\mathbf{p}^{(0)},\ \mathbf{p}^{(3)} = \mathbf{P}^T\mathbf{p}^{(2)} = \left[\mathbf{P}^T\right]^3\mathbf{p}^{(0)},\ \ldots,$$

$$\mathbf{p}^{(r)} = \mathbf{P}^T\mathbf{p}^{(r-1)} = \left[\mathbf{P}^T\right]^r\mathbf{p}^{(0)}.$$

Pod vjerovatnoćom u stacionarnom stanju podrazumijevamo da se vjerovatnoće u uzastopnim stanjima ne mijenjaju i da dalje množenje vjerovatnoća transponovanom matricom tranzicije $\mathbf{P}^T$ ne mijenja dalje vjerovatnoće stanja:

$$\mathbf{p} = \mathbf{P}^T\mathbf{p}.$$

Dalje slijedi:

$$\mathbf{Ip} = \mathbf{P}^T\mathbf{p} \Rightarrow [\mathbf{P}^T - \mathbf{I}]\mathbf{p} = \mathbf{0},$$

gdje smo sa **I** označili jediničnu matricu odgovarajućih dimenzija. Da bismo pojasnili, prikažimo ovaj sistem u razvijenoj formi za konkretni slučaj:

$$\begin{bmatrix} P(a\,|\,a)-1 & P(a\,|\,b) & P(a\,|\,c) \\ P(b\,|\,a) & P(b\,|\,b)-1 & P(b\,|\,c) \\ P(c\,|\,a) & P(c\,|\,b) & P(c\,|\,c)-1 \end{bmatrix} \begin{bmatrix} p(a) \\ p(b) \\ p(c) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Dobijen je homogeni (nula vektor je na desnoj strani) sistem tri jednačine sa tri nepoznate. Moguće rješenje ovog sistema, ako su jednačine nezavisne, jeste nula vektor $[p(a), p(b), p(c)] = [0\ 0\ 0]$. Ovo trivijalno rješenje nije ono što nam odgovara kada imamo na umu da su rješenja vjerovatnoće i da bi suma vjerovatnoća simbola alfabeta morala biti jednaka 1. Stoga, jednu od ove tri jednačine treba da zamijenimo jednačinom (ograničenjem) $p(a) + p(b) + p(c) = 1$, pa se tek onda, rješavanjem modifikovanog sistema jednačina, mogu dobiti vjerovatnoće u stacionarnom stanju:

$$\begin{bmatrix} P(a\,|\,a)-1 & P(a\,|\,b) & P(a\,|\,c) \\ P(b\,|\,a) & P(b\,|\,b)-1 & P(b\,|\,c) \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} p(a) \\ p(b) \\ p(c) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{P}'\mathbf{p} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

gdje je **P'** modifikovana transponovana tranziciona matrica, dobijena zamjenom jedne vrste s uslovom $p(a)+p(b)+p(c)=1$. Rješenje sistema jednačina, odnosno vjerovatnoće u stacionarnom stanju dobijaju se kao:

$$\mathbf{p}=[\mathbf{P}']^{-1}\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T.$$

U konkretnom slučaju grafa tranzicije sa Slike II.4, sa vrijednostima datim u Primjeru II.2, dobijamo:

$$\mathbf{p}=\begin{bmatrix}-2/3 & 1/4 & 1/4\\ 1/3 & -1/2 & 1/4\\ 1 & 1 & 1\end{bmatrix}^{-1}\begin{bmatrix}0\\0\\1\end{bmatrix}.$$

Dobijamo rezultat $\mathbf{p}=[3/11\ 4/11\ 4/11]$, to jest $p(a)=3/11$, $p(b)=p(c)=4/11$. Napomenimo da je za invertibilnost matrice **P'** potrebno da je ona ranga 3, a to znači da tranziciona matrica **P** mora biti ranga 2 (da ima dvije nezavisne kolone i vrste). U slučaju da ne raspolažete softverom koji računa inverznu matricu ili da je ona velika, pa se predmetna operacija ne može provesti ručno, može se, naravno, koristiti gausovska eliminacija, koju u edukativne svrhe ponavljamo za predmetni primjer:

$$\begin{aligned}-\frac{2}{3}p(a)+\frac{1}{4}p(b)+\frac{1}{4}p(c)&=0\\ \frac{1}{3}p(a)-\frac{1}{2}p(b)+\frac{1}{4}p(c)&=0\\ p(a)+p(b)+p(c)&=1.\end{aligned}$$

Oduzimanjem prve i druge relacije slijedi da je:

$$-p(a)+\frac{3}{4}p(b)=0\Rightarrow p(a)=\frac{3}{4}p(b).$$

Uvrštavanjem ovog izraza u prvu jednačinu, dobijamo:

$$-\frac{2}{3}\frac{3}{4}p(b)+\frac{1}{4}p(b)+\frac{1}{4}p(c)=0\Rightarrow-\frac{1}{4}p(b)+\frac{1}{4}p(c)=0\Rightarrow p(b)=p(c).$$

Ovo je bilo i očigledno jer su u datom slučaju čvorovi *b* i *c* potpuno simetrični, ali smo ovdje uradili postepeno radi uvježbavanja elementarne aritmetike. Sada zamijenimo dobijene izraze za $p(a)$ i $p(c)$ u treću relaciju, pa dobijamo:

$$\frac{3}{4}p(b)+p(b)+p(b)=1\Rightarrow\frac{11}{4}p(b)=1\Rightarrow p(b)=\frac{4}{11}$$

$$p(c)=\frac{4}{11}\qquad p(a)=\frac{3}{11}\qquad p(b)=\frac{4}{11}.$$

čime su dobijeni izrazi za vjerovatnoću u stacionarnom stanju. Ovaj jednostavni sistem biće korišćen u narednom poglavlju radi ilustracije određenih tehnika za kodiranje izvora.□

Primjer II.5. Odrediti vjerovatnoće u stacionarnom stanju za Markovljev sistem drugog reda sa binarnim alfabetom, s uslovnim vjerovatnoćama iz Primjera II.3.

Rješenje: Predmetni zadatak se može riješiti u nekoliko poteza ako uočimo da su dešavanja vezana za bitove 0 i 1 simetrična.Zaključujemo da su vjerovatnoće ovih simbola u stacionarnom stanju iste $p(0)=p(1)=1/2$. Međutim, radi uvježbavanja kako se predmetne operacije vrše, idemo korak po korak.

Uzmimo da je vjerovatnoća bitova u početnom trenutku $p'(0)$ i $p'(1)$. Pretpostavimo da znamo uslovne vjerovatnoće u početnom trenutku $p'(0|0)$, $p'(1|0)$, $p'(0|1)$, $p'(1|1)$. Združene vjerovatnoće pojedinih parova početnih simbola su:

$$p'(00)=p'(0)p'(0|0) \qquad p'(10)=p'(0)p'(1|0)$$
$$p'(01)=p'(1)p'(0|1) \qquad p'(11)=p'(1)p'(1|1).$$

Sada možemo da odredimo kolike su združene vjerovatnoće u narednom trenutku:

$$\begin{bmatrix} p''(00) \\ p''(01) \\ p''(10) \\ p''(11) \end{bmatrix} = \begin{bmatrix} 0.7 & 0 & 0.5 & 0 \\ 0.3 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.3 \\ 0 & 0.5 & 0 & 0.7 \end{bmatrix} \begin{bmatrix} p'(00) \\ p'(01) \\ p'(10) \\ p'(11) \end{bmatrix}.$$

Za stacionarno stanje važi jednačina:

$$\begin{bmatrix} p(00) \\ p(01) \\ p(10) \\ p(11) \end{bmatrix} = \begin{bmatrix} 0.7 & 0 & 0.5 & 0 \\ 0.3 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.3 \\ 0 & 0.5 & 0 & 0.7 \end{bmatrix} \begin{bmatrix} p(00) \\ p(01) \\ p(10) \\ p(11) \end{bmatrix},$$

odnosno:

$$\begin{bmatrix} -0.3 & 0 & 0.5 & 0 \\ 0.3 & -1 & 0.5 & 0 \\ 0 & 0.5 & -1 & 0.3 \\ 0 & 0.5 & 0 & -0.3 \end{bmatrix} \begin{bmatrix} p(00) \\ p(01) \\ p(10) \\ p(11) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Ponovo jednu jednačinu (recimo, posljednju) možemo zamijeniti uslovom $p(00)+p(01)+p(10)+p(11)=1$:

$$\begin{bmatrix} -0.3 & 0 & 0.5 & 0 \\ 0.3 & -1 & 0.5 & 0 \\ 0 & 0.5 & -1 & 0.3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} p(00) \\ p(01) \\ p(10) \\ p(11) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Rješavanjem ovog sistema dobijamo da su vjerovatnoće združenih stanja u stacionarnom stanju $p(00) = p(11) = 1/3.2 = 5/16$, dok je $p(10) = p(01) = 3/16$. Dalje, marginalne vjerovatnoće u stacionarnom stanju $p(0)$ i $p(1)$ su:

$$p(0) = p(00) + p(10) = 0.5$$
$$p(1) = p(10) + p(11) = 0.5.$$

II.3 Zadaci i softverska realizacija

II.3.1 Riješeni zadaci

2.1. Uspostaviti vezu između $H(Y|X)$ i $H(Y|X=x)$.

Rješenje:

$$H(Y \mid X) = -\sum_x \sum_y p(x,y) \log p(y \mid x) = -\sum_x \sum_y p(y \mid x) p(x) \log p(y \mid x)$$
$$= \sum_x p(x) \left[-\sum_y p(y \mid x) \log p(y \mid x) \right] = \sum_x p(x) H(Y \mid X = x).$$

2.2. Dokazati da ako je $f(x)$ konveksna funkcija i X slučajna promjenljiva, tada važi:

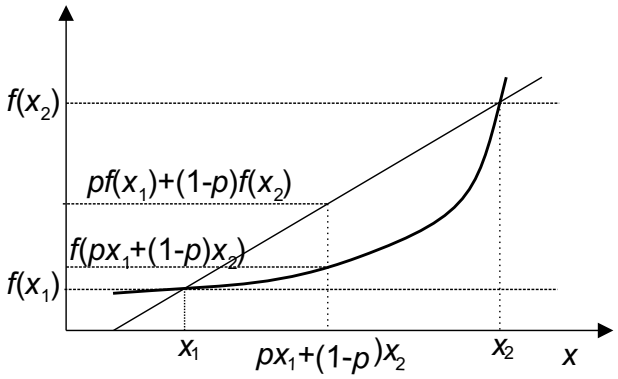
$$\mathrm{E}[f(X)] \geq f(\mathrm{E}[X])$$

Ako je funkcija striktno konveksna, tada je $X = \mathrm{E}[X]$ zadovoljeno sa vjerovatnoćom 1, odnosno X je konstantno.

Rješenje: Predmetni iskaz naziva se Jensenovom nejednakošću. Za one koji ne znaju pojam **konveksnosti**, dajemo sljedeću definiciju.

Definicija II.9: Funkcija $f(x)$ je konveksna na intervalu (a,b) ako za svako x_1, $x_2 \in (a,b)$ i $0 \leq p \leq 1$ važi $f(px_1 + (1-p)x_2) \leq pf(x_1) + (1-p)f(x_2)$. Funkcija je striktno konveksna ako jednakost važi samo za $p = 0$ i $p = 1$.

Na Slici II.8 prikazana je konveksna (udubljena) funkcija na intervalu (a,b). Suprotno od konveksne funkcije je **konkavna** (ispupčena) funkcija. Napominjemo da prevod ovih termina sa latinskog govori nešto drugo o prirodi ovih funkcija, ali to je, uostalom, vezano za pravac iz kojeg se gleda na udubljenost ili ispupčenost.



Slika II.8. Ilustracija konveksne funkcije

Pređimo sada na dokaz nejednakosti. Lijeva strana nejednakosti može da se izrazi (za diskretne slučajne promjenljive) kao:

$$\mathrm{E}[f(X)] = \sum_{i=1}^{k} p_i f(x_i),$$

gdje slučajna promjenljiva uzima k diskretnih vrijednosti x_i, $i = 1,...,k$, sa vjerovatnoćama p_i, $i = 1,...,k$. Dalje, možemo pisati:

$$\mathrm{E}[f(X)] = \sum_{i=1}^{k} p_i f(x_i) = \sum_{i=1}^{k-1} p_i f(x_i) + p_k f(x_k) = (1-p_k)\sum_{i=1}^{k-1} \frac{p_i}{1-p_k} f(x_i) + p_k f(x_k) =$$

$$= p_k f(x_k) + (1-p_k)\sum_{i=1}^{N-1} p_i' f(x_i),$$

gdje je $p_i' = p_i / (1-p_k)$. Drugi član u izrazu može se zapisati korišćenjem osobine konveksnosti:

$$\sum_{i=1}^{N-1} p_i' f(x_i) \geq f\left(\sum_{i=1}^{N-1} p_i' x_i\right).$$

Dalje se dobija da je:

$$\mathrm{E}[f(X)] \geq p_k f(x_k) + (1-p_k) f\left(\sum_{i=1}^{k-1} p_i' x_i\right) \geq$$

$$\geq f\left(p_k f(x_k) + (1-p_k)\sum_{i=1}^{k-1} p_i' x_i\right) = f\left(\sum_{i=1}^{k} p_i x_i\right) = f\left(\mathrm{E}[X]\right)$$

čime je teorema dokazana.

2.3. Na osnovu Jensenove nejednakosti dokazati da za sve nenegativne brojeve $(a_1,...,a_n)$ i $(b_1,...,b_n)$ važi:

$$\sum_{i=1}^{n}\left(a_i \log\frac{a_i}{b_i}\right) \geq \sum_{i=1}^{n} a_i \log\left(\sum_{i=1}^{n} a_i \Big/ \sum_{i=1}^{n} b_i\right),$$

dok jednakost važi za $a_i = b_i$. Po konvenciji se usvaja da je $0\log 0 = 0$, $a\log(a/0) = \infty$ i $0\log(0/0) = 0$.

Rješenje: Funkcija $f(x) = x\log x$ je striktno konveksna za svako $0 \leq x < \infty$ jer je:

$$f'(x) = \log x + x\frac{1}{x}\log e = \log x + \log e \qquad f''(x) = \frac{1}{x}\log e > 0$$

$$\sum_{i=1}^{n} p_i x_i \log x_i = \mathrm{E}[X\log X] \geq \mathrm{E}[X]\log(\mathrm{E}[X]) = \left(\sum_{i=1}^{n} p_i x_i\right)\log\left(\sum_{i=1}^{n} p_i x_i\right),$$

gdje je p_i neka funkcija koja ispunjava uslove funkcije gustine raspodjele. Uzimajući da je:

$$p_i = b_i / \sum_{i=1}^{n} b_i \qquad x_i = a_i / \sum_{i=1}^{n} a_i,$$

može se pisati:

$$\sum_{i=1}^{n} \frac{b_i}{\sum_{i=1}^{n} b_i}\frac{a_i}{b_i}\log\left(\frac{a_i}{b_i}\right) \geq \left(\sum_{i=1}^{n} \frac{b_i}{\sum_{i=1}^{n} b_i}\frac{a_i}{b_i}\right)\log\left(\sum_{i=1}^{n} \frac{b_i}{\sum_{i=1}^{n} b_i}\frac{a_i}{b_i}\right)$$

$$\sum_{i=1}^{n}\left(\frac{a_i}{\sum_{i=1}^{n} b_i}\log\frac{a_i}{b_i}\right) \geq \left(\sum_{i=1}^{n} \frac{a_i}{\sum_{i=1}^{n} b_i}\right)\log\left(\sum_{i=1}^{n} \frac{a_i}{\sum_{i=1}^{n} b_i}\right)$$

$$\frac{\sum_{i=1}^{n} a_i \log\frac{a_i}{b_i}}{\sum_{i=1}^{n} b_i} \geq \left(\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}\right)\log\left(\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}\right).$$

Na kraju treba pomnožiti obje strane nejednačine sa $\sum_{i=1}^{n} b_i$, čime je dokaz završen.

2.4. Na osnovu prethodnih zadataka dokazati da je Kalbek-Lejblerova distanca nenegativna veličina.

Rješenje: Na osnovu rješenja zadatka 2.2, možemo pisati:

$$D(p\,\|\,q)=\sum_x p(x)\log\frac{p(x)}{q(x)}\geq\sum_x p(x)\log\frac{\sum_x p(x)}{\sum_x q(x)}=1\log 1=0.$$

2.5. Skicirati entropiju ternarnog događaja. Uzeti da su vjerovatnoće pojedinih događaja p, q, i $1-p-q$, pa entropiju crtati u zavisnosti od p i q. Kada je ova entropija maksimalna i koliko iznosi?

Rješenje: Entropija iznosi:

$$H(p,q)=-p\log p-q\log q-(1-p-q)\log(1-p-q).$$

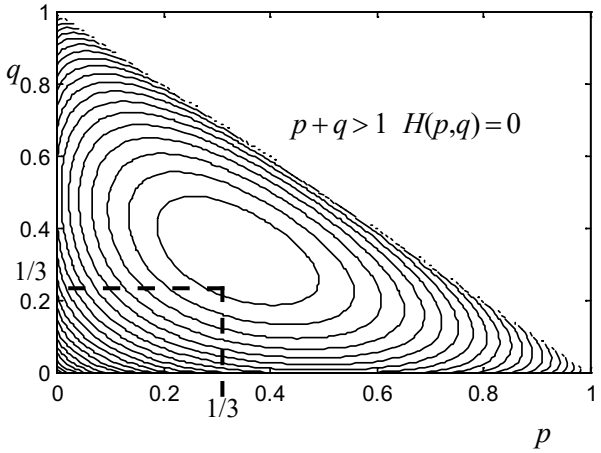
Entropija se može sračunati samo pod ograničenjem da je $p+q\leq 1$. Ilustracija entropije je data na slici u konturnom crtežu sa izolinijama (Slika II.9, sa podebljanom linijom koja omeđava domen u kome entropija ternarnog događaja nije definisana). I u ovom slučaju se maksimalna entropija dobija kada je $p=q=1-p-q=1/3$. Dokazuje se određivanjem izvoda funkcije $H(p,q)$ po p i po q i njihovim izjednačavanjem sa nulom. Ova veličina iznosi $H(1/3,1/3)=1.585$ bita.

2.6. Entropija skupa X je $H(X)=8$ bita, a entropija skupa Y je $H(Y)=12$ bita. U kojim se granicama nalazi $H(Y|X)$ ako se $H(X|Y)$ mijenja od minimalne do maksimalne vrijednosti?

Rješenje: Međusobna informacije se može zapisati kao:

$$I(X;Y)=H(X)-H(X\,|\,Y)\qquad I(X;Y)=H(Y)-H(Y\,|\,X)$$

$$H(Y\,|\,X)=H(Y)-H(X)+H(X\,|\,Y).$$



Slika II.9. Izolinije (konturni dijagram) entropije ternarnog alfabeta

Dalje važi:

$$H(Y \mid X)_{\max} = H(Y) - H(X) + H(X \mid Y)_{\max}$$
$$H(Y \mid X)_{\min} = H(Y) - H(X) + H(X \mid Y)_{\min}.$$

Maksimalno $H(X|Y)$ je jednako $H(X)$, pa je $H(Y \mid X)_{\max} = H(Y) = 12$ bita, dok je minimalno $H(X|Y)$ jednako 0, pa se dobija: $H(Y \mid X)_{\max} = H(Y) - H(X) = 4$ bita.

2.7. Neka su vjerovatnoće simbola 1/4 i 3/4. Kolika je entropija ovog sistema? Kolika je entropija sistema kod kojeg se šalju kombinacije od po dva simbola sa datim vjerovatnoćama ako su simboli međusobno nezavisni? Ponoviti proceduru za tri nezavisna simbola. Koliku entropiju očekujete kod *n* nezavisnih ponavljanja simbola po istim pravilima?

Rješenje: Entropija događaja je jednaka:

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p) = 0.8113 \text{ bita.}$$

Združeni događaj pojavljivanja dva simbola ima ukupno 4 moguće pojave sa vjerovatnoćama: 1/16 (recimo, 00), 3/16 (01), 3/16 (10) i 9/16 (11). Entropija ovog alfabeta (sa simbolima {00, 01, 10, 11}) je:

$$H_2 = 2H(p) = 1.6226 \text{ bita.}$$

Pošto se radi o međusobno nezavisnim simbolima, imamo da je entropija uslovnog događaja jednaka nuli, to jest da je entropija združenog događaja jednaka sumi entropija pojedinačnih događaja, pa važi:

$$H_3 = 3H(p) = 2.4339 \text{ bita.}$$

Za *k* uzastopnih međusobno nezavisnih simbola sa datom raspodjelom, važi:

$$H_k = kH(p).$$

2.8. Bacaju se dvije kocke. X je binarni događaj koji daje 1 ako je vrijednost bačena sa prvom kockom veća od vrijednosti bačene drugom kockom, dok je događaj Y binarni događaj koji je jednak 1 ako je suma na dvije kocke parna. Odrediti pojedine vjerovatnoće događaja X, Y i združenog događaja X i Y, kao i uslovne vjerovatnoće. Odrediti entropije i međusobnu informaciju.

Rješenje: Odredićemo prvo združene vjerovatnoće dva događaja. Vjerovatnoća da se dogodi $X=0$ i $Y=0$ podrazumijeva situacije da su se pri bacanju kocke dogodili (prva kocka daje manji ili jednak broj drugoj i neparnu sumu){(1,2), (1,4), (1,6), (2,3), (2,5), (3,4), (3,6), (4,5), (5,6)} (prva vrijednost u uređenom paru je ishod sa prve kocke, dok je drugi ishod sa druge kocke). Dakle, imamo ukupno 9 situacija i vjerovatnoću $p(X=0, Y=0) = 9/36 = 1/4$. Drugi združeni događaj $X=0$ i $Y=1$ (prva

$p(x,y)$	$Y=0$	$Y=1$	$p(x)$
$X=0$	1/4	1/3	7/12
$X=1$	1/4	1/6	5/12
$p(y)$	1/2	1/2	

Tabela II.1. Združena i marginalne vjerovatnoće za događaje X i Y

$p(x\|y)$	$Y=0$	$Y=1$
$X=0$	1/2	2/3
$X=1$	1/2	1/3

$p(y\|x)$	$Y=0$	$Y=1$
$X=0$	3/7	4/7
$X=1$	3/5	2/5

Tabela II.2. Uslovne vjerovatnoće za događaje X i Y

kocka daje manju ili jednaku vrijednost drugoj, a suma na kockama je parna) dešava se u sljedećim situacijama: {(1,1), (1,3), (1,5), (2,2), (2,4), (2,6), (3,3), (3,5), (4,4), (4,6), (5,5), (6,6)}, pa je združena vjerovatnoća jednaka $p(X=0,Y=1)=12/36=1/3$. Naredni združeni događaj je $X=1$ i $Y=0$ s ishodima {(2,1), (3,2), (4,1), (4,3), (5,2), (5,4), (6,1), (6,3), (6,5)}, pa je vjerovatnoća $p(X=1,Y=0)=9/36=1/4$. Preostaje još samo događaj $X=1$ i $Y=1$, koji daju ishodi bacanja kocke {(3,1), (4,2), (5,1), (5,3), (6,2), (6,4)} sa vjerovatnoćom $p(X=1,Y=1)=6/36=1/6$. U Tabeli II.1 prikazujemo ove združene vjerovatnoće s odgovarajućim marginalnim, dok naredne tabele prikazuju uslovne vjerovatnoće.

Očigledno je $H(Y)=1$ bit, dok je

$$H(X)=-\frac{5}{12}\log_2\frac{5}{12}-\frac{7}{12}\log_2\frac{7}{12}\approx 0.9799 \text{ bita}.$$

Združena entropija je:

$$H(X,Y)=-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{3}\log_2\frac{1}{3}-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{6}\log_2\frac{1}{6}\approx 1.9591 \text{ bita}.$$

Uslovne entropije su sada:

$$H(X\mid Y)=H(X,Y)-H(Y)\approx 0.9792 \text{ bita}$$

$$H(Y\mid X)=H(X,Y)-H(X)\approx 0.9591 \text{ bita}.$$

Konačno, međusobna informacija je jednaka:

$$I(X;Y)=H(X)+H(Y)-H(X,Y)\approx 0.0208 \text{ bita}.$$

Premda se, možda, na prvi pogled ne bi tako reklo, ova dva događaja imaju veoma malu međusobnu informaciju, odnosno malo govore jedan o drugom.

2.9. $D(p\|q)$ je konveksna za par funkcija gustine raspodjele (p,q) ako su (p_1,q_1) i (p_2,q_2) funkcije gustine raspodjele:

$$D(\mu p_1 + (1-\mu)p_2) \,\|\, \mu q_1 + (1-\mu)q_2) \le \mu D(p_1 \,\|\, q_1) + (1-\mu)D(p_2 \,\|\, q_2)$$

za svako $0 \le \mu \le 1$.

Rješenje: Pođimo od definicije Kalbek–Lejblerove funkcije:

$$D(p \,\|\, q) = \sum_{x \in A} p(x) \log \frac{p(x)}{q(x)}.$$

Lijeva strana izraza u razvijenoj formi je:

$$D(\mu p_1 + (1-\mu)p_2) \,\|\, \mu q_1 + (1-\mu)q_2) = \sum_{x \in A} [\mu p_1(x) + (1-\mu)p_2(x)] \log \frac{[\mu p_1(x) + (1-\mu)p_2(x)]}{[\mu q_1(x) + (1-\mu)q_2(x)]}$$

dok je desna strana:

$$\mu D(p_1 \,\|\, q_1) + (1-\mu)D(p_2 \,\|\, q_2) = \mu \sum_{x \in A} p_1(x) \log \frac{p_1(x)}{q_1(x)} + (1-\mu) \sum_{x \in A} p_2(x) \log \frac{p_2(x)}{q_2(x)}.$$

Upotrijebimo sada relaciju iz zadatka 2.3

$$\sum_{i=1}^{n} \left(a_i \log \frac{a_i}{b_i} \right) \ge \sum_{i=1}^{n} a_i \log \left(\sum_{i=1}^{n} a_i \Big/ \sum_{i=1}^{n} b_i \right).$$

Ako je $a_i = \lambda_i p_i$, $b_i = \lambda_i p_i$ i $\lambda_1 = \mu$ i $\lambda_2 = 1 - \mu$, dobijamo:

$$\sum_{i=1}^{2} (\lambda_i p_i(x)) \log \frac{\sum_{i=1}^{2} (\lambda_i p_i(x))}{\sum_{i=1}^{2} (\lambda_i q_i(x))} \le \sum_{i=1}^{2} \lambda_i p_i(x) \log \frac{\lambda_i p_i(x)}{\lambda_i q_i(x)}$$

$$[\mu p_1(x) + (1-\mu)p_2(x)] \log \frac{[\mu p_1(x) + (1-\mu)p_2(x)]}{[\mu q_1(x) + (1-\mu)q_2(x)]} \le$$

$$\le \mu p_1(x) \log \frac{p_1(x)}{q_1(x)} + (1-\mu)p_2(x) \log \frac{p_2(x)}{q_2(x)}.$$

Ako sada obje strane sumiramo po elementima skupa A, dobijamo:

$$\sum_{x \in A} [\mu p_1(x) + (1-\mu)p_2(x)] \log \frac{[\mu p_1(x) + (1-\mu)p_2(x)]}{[\mu q_1(x) + (1-\mu)q_2(x)]} \le$$

$$\le \mu \sum_{x \in A} p_1(x) \log \frac{p_1(x)}{q_1(x)} + (1-\mu) \sum_{x \in A} p_2(x) \log \frac{p_2(x)}{q_2(x)}.$$

Čime smo zapravo kompletirali dokaz:

$$D(\mu p_1 + (1-\mu)p_2) \,||\, \mu q_1 + (1-\mu)q_2) \le \mu D(p_1 \,||\, q_1) + (1-\mu)D(p_2 \,||\, q_2).$$

2.10. Teniski meč na Grend slem turnirima igra se do 3 dobijena seta. Vjerovatnoća da teniser 1 dobije set protiv tenisera broj 2 jednaka je 0.7. Postoje dva događaja: A − broj setova odigranih u meču i B koji je 1 ako dobija prvi teniser i 0 ako meč dobija drugi teniser. Odrediti entropije ovih događaja, združenu entropiju, uslovne entropije i međusobnu informaciju.

Rješenje: Formirajmo tabelu združene i marginalnih vjerovatnoća (Tabela II.3).

Združena entropija je data kao:

$$H(A,B) = -0.343\log_2 0.343 - 0.027\log_2 0.027 - 0.3087\log_2 0.3087 - \\ - 0.0567\log_2 0.0567 - 0.18522\log_2 0.18522 - 0.07938\log_2 0.07938 \approx \\ \approx 2.1691 \text{ bita.}$$

Entropije marginalnih događaja su:

$$H(A) = -0.37\log_2 0.37 - 0.3654\log_2 0.03654 - 0.2646\log_2 0.2646 \approx 1.5690 \text{ bita}$$

$$H(B) = -0.83692\log_2 0.83692 - 0.16308\log_2 0.16308 \approx 0.6416 \text{ bita}.$$

Uslovne entropije su:

$$H(A \mid B) = H(A,B) - H(B) \approx 0.6002 \text{ bita}$$

$$H(B \mid A) = H(A,B) - H(A) \approx 1.5275 \text{ bita}.$$

Konačno, međusobna informacija je:

$$I(A;B) = H(A) + H(B) - H(A,B) \approx 0.0415 \text{ bita}.$$

2.11. Šaljemo tri bita koji su međusobno nezavisni i na isti način distribuirani, s vjerovatnoćom pojave bita 1 jednakom p. Na osnovu ove sekvence formiramo dva događaja (alfabeta): X, koji predstavlja broj jedinica u sekvenci, i Y, koji je jednak 1 ako su prvi i treći bit u sekvenci isti i 0 ako su različiti. Odrediti entropije događaja, združenog događaja, uslovnih događaja i međusobnu informaciju.

A \ B	1	0	$p(A)$
3	$p^3 = 0.7^3 = 0.343$	$q^3 = 0.3^3 = 0.027$	0.37
4	$3p^3q = 0.3087$	$3q^3p = 0.0567$	0.3654
5	$6p^3q^2 = 0.18522$	$6p^2q^3 = 0.07938$	0.2646
$p(B)$	0.83692	0.16308	

Tabela II.3. Združena i marginalne vjerovatnoće za slučaj događaja izvedenih iz ishoda teniskog meča

Rješenje: Kao i kod prethodnih zadataka, možda je najlakše formirati tabelu združene i marginalnih vjerovatnoća (Tabela II.4).

Entropija marginalnog događaja $H(Y)$ je entropija binarnog događaja s vjerovatnoćom $2p(1-p)$:

$$H(Y) = H(2p(1-p)).$$

Znatno složenije je određivanje ostalih entropija:

$$\begin{aligned}
H(X,Y) &= -(1-p)^3 \log_2 (1-p)^3 - p(1-p)^2 \log_2 p(1-p)^2 - 2p(1-p)^2 \log_2 2p(1-p)^2 - \\
&\quad - p^2(1-p)\log_2 p^2(1-p) - 2p^2(1-p)\log_2 2p^2(1-p) - p^3 \log_2 p^3 = \\
&= -3p^3 \log_2 p - p(1-p)^2 \log_2 p - 2p(1-p)^2 \log_2 (1-p) \\
&\quad - 2p(1-p)^2 \log_2 p - 4p(1-p)^2 \log_2 (1-p) - 2p(1-p)^2 \\
&\quad - 2p^2(1-p)\log_2 p \\
&\quad - 2p(1-p)^2 \log_2 (1-p) - 2p^2(1-p) - 4p^2(1-p)\log_2 p \\
&\quad - 2p^2(1-p)\log_2 (1-p) - 3(1-p)^3 \log_2 (1-p) = \\
&= -[3p^3 + 3p(1-p)^2 + 6p^2(1-p)]\log_2 p \\
&\quad - [6p(1-p)^2 + 3p^2(1-p) + 3(1-p)^3]\log_2 (1-p) - 2p(1-p)^2 - 2p^2(1-p) = \\
&= -3p[p^2 + 2p(1-p) + (1-p)^2]\log_2 p \\
&\quad - 3(1-p)[p^2 + 2p(1-p) + (1-p)^2]\log_2 (1-p) - 2p(1-p) = \\
&= -3p\log_2 - 3(1-p)\log_2 (1-p) - 2p(1-p) = 3H(p) - 2p(1-p).
\end{aligned}$$

Na sličan način se određuje entropija događaja X:

$$\begin{aligned}
H(X) &= -p^3 \log_2 p^3 - 3p^2(1-p)\log_2 [3p^2(1-p)] \\
&\quad - 3p(1-p)^2 \log_2 [3p(1-p)^2 [-(1-p)^3 \log_2 (1-p)^3 = \\
&= -3p^3 \log_2 p - 3p^2(1-p)\log_2 3 - 6p^2(1-p)\log_2 p \\
&\quad - 3p^2(1-p)\log_2 (1-p) - 3p(1-p)^2 \log_2 3 - \\
&\quad - 3p(1-p)^2 \log_2 p - 6p(1-p)^2 \log_2 (1-p) - \\
&\quad - 3(1-p)^3 \log_2 (1-p) = \\
&= [-3p^3 - 6p^2(1-p) - 3p(1-p)^2]\log_2 p - \\
&\quad + [-3p^2(1-p) - 6p(1-p)^2 - 3(1-p)^3]\log_2 (1-p) - \\
&\quad - 3p(1-p)^2 \log_2 3 - 3p^2(1-p)^2 \log_2 3 = \\
&= -3p\log_2 p - 3(1-p)\log_2 (1-p) \\
&\quad - 3p(1-p)\log_2 3[p + (1-p)] = 3H(p) - 3p(1-p)\log_2 3.
\end{aligned}$$

X \ Y	1	0	$p(X)$
0	$(1-p)^3$	0	$(1-p)^3$
1	$p(1-p)^2$	$2p(1-p)^2$	$3p(1-p)^2$
2	$p^2(1-p)$	$2p^2(1-p)$	$3p^2(1-p)$
3	p^3	0	p^3
$p(Y)$	$1-2p(1-p)$	$2p(1-p)$	

Tabela II.4. Združena i marginalne vjerovatnoće za slučaj događaja izvedenih iz trobitne sekvence

Sada možemo da odredimo uslovne entropije:

$$H(X \mid Y) = H(X,Y) - H(Y) = 3H(p) - 2p(1-p) - H(2p(1-p))$$

$$H(Y \mid X) = H(X,Y) - H(X) = 3H(p) - 2p(1-p) - 3H(p) + 3p(1-p)\log_2 3 =$$

$$= p(1-p)[3\log_2 3 - 2] = p(1-p)[\log_2 3^3 - \log_2 2^2] = p(1-p)\log_2 \frac{27}{4}.$$

Konačno, međusobna informacija je:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) =$$

$$= 3H(p) - 3p(1-p)\log_2 3 + H(2p(1-p)) - 3H(\boldsymbol{p}) + 2p(1-p) =$$

$$= H(2p(1-p)) - p(1-p)\log_2 \frac{27}{4}.$$

2.12. Posjedujemo jedan novčić od 1 euro i dva novčića od 2 eura. Jedan od novčića od 2 eura nije fer, odnosno pada na stranu na kojoj je upisana vrijednost s vjerovatnoćom 0.25. Ostala dva novčića su fer, odnosno padaju na obje strane s podjednakim šansama. Događaj A je suma vrijednosti novčića koji padnu na stranu s upisanom vrijednošću, a događaj B je broj novčića koji padnu na tu stranu. Odrediti entropije ovih događaja, združenu entropiju, uslovne entropije i međusobnu informaciju.

Rješenje: Kao i u prethodnim primjerima, kreirajmo tabelu sa združenom i marginalnim vjerovatnoćama.

Združena entropija je:

$$H(A,B) = -2 \cdot 0.1875\log_2 0.1875 - 2 \cdot 0.25\log_2 0.25 - 2 \cdot 0.0625\log_2 0.0625 \approx$$

$$\approx 2.4056 \text{ bita}$$

$$H(A) = H(A,B) \approx 2.4056 \text{ bita}$$

$$H(B) = -0.1875\log_2 0.1875 - 0.4375\log_2 0.4375$$

$$- 0.3125\log_2 0.3125 - 0.0625\log_2 0.0625 \approx 1.749 \text{ bita}.$$

Uslovna entropija $H(B|A)$ je očigledno nula ($H(B|A)=0$), jer kada znamo sumu bacanja novčića znamo i koliko je novčića palo na odgovarajuću stranu. S druge strane, uslovna entropija $H(A|B)$ je nenulta, jer kada znamo koliko novčića je palo na neku stranu i dalje nismo sasvim sigurni u zbir. Ova uslovna entropija iznosi:

$$H(A\,|\,B)=H(A,B)-H(B)\approx 0.6566 \text{ bita}.$$

Međusobna informacija je onda jednaka entropiji događaja $H(B)$ (koja je podskup entropije združenog događaja i entropije događaja $H(A)$):

$$I(A;B)=H(A)+H(B)-H(A,B)=H(A)+H(B)-H(A)=H(B)\approx 1.749 \text{ bita}.$$

2.13. U kutiji se nalaze tri bijele i tri crne kuglice. Izvlače se redom kuglice. Proces izvlačenja se zaustavlja kada se zaredom izvuku kuglice iste boje. Događaj A je broj izvučenih kuglica, dok je događaj B boja posljednje izvučene kuglice. Odrediti entropije ovih događaja, združenu entropiju, uslovne entropije i međusobnu informaciju.

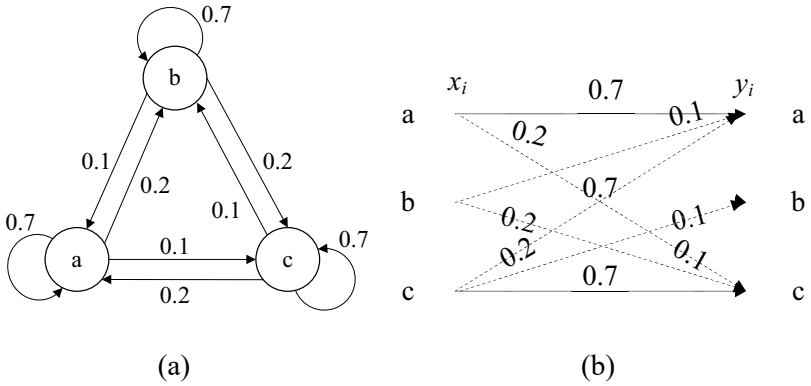
Rješenje: Ponovo formirajmo tabelu sa vjerovatnoćama. Uzmimo da je, kod događaja B, bijela boja prikazana kao broj 1, dok je crna broj 0. Potrebno je minimalno dva izvlačenja, a može se dogoditi da je potrebno da se izvuče svih šest kuglica. Odgovarajuće vjerovatnoće prikazane su u Tabeli II.6.

A \ B	0	1	2	3	$p(A)$
0	$0.5^2\cdot 0.75=0.1875$	0	0	0	0.1875
1	0	0.1875	0	0	0.1875
2	0	$0.5\cdot 0.5\cdot 0.75+0.5\cdot 0.5\cdot$ $\cdot 0.25=0.25$	0	0	0.25
3	0	0	0.25	0	0.25
4	0	0	0.0625	0	0.0625
5	0	0	0	0.0625	0.0625
$p(B)$	0.1875	0.4375	0.3125	0.0625	

Tabela II.5. Združena i marginalne vjerovatnoće za slučaj događaja izvedenih iz bacanja novčića

A \ B	0	1	$p(A)$
2	$0.5\cdot 0.4=0.2$	0.2	0.4
3	$0.5\cdot 0.6\cdot 0.5=0.15$	0.15	0.3
4	0.05	0.05	0.1
5	0.05	0.05	0.1
6	0.05	0.05	0.1
$p(B)$	0.5	0.5	

Tabela II.6. Združena i marginalne vjerovatnoće događaja izvlačenja kuglica



Slika II.10. Markovljev sistem prvog reda: (a) Graf tranzicije; (b) Trelis dijagram

Združena entropija je:

$$H(A,B) = -2 \cdot 0.2 \log_2 0.2 - 2 \cdot 0.15 \log_2 0.15 - 6 \cdot 0.05 \log_2 0.05 \approx 3.0464 \text{ bita}.$$

Entropija događaja B je očigledno $H(B) = 1$, dok važi:

$$H(A) \approx 2.0464 \text{ bita}.$$

Uslovne entropije su:

$$H(A|B) = H(A,B) - H(B) = H(A) = 2.0464 \text{ bita}.$$

Dakle, informacija o događaju B ni na koji način ne umanjuje entropiju događaja A. Isto važi i u suprotnom smjeru jer je:

$$H(B|A) = H(A,B) - H(A) = H(B) = 1 \text{ bit}.$$

Iz ovoga slijedi da je međusobna informacija, koju dijele ova dva događaja, jednaka nuli: $I(A;B) = 0$ bita.

2.14. Dat je Markovljev sistem I reda s alfabetom $X = \{a,b,c\}$ i uslovnim vjerovatnoćama datim matricom:

$$\begin{bmatrix} 0.7 & 0.1 & 0.2 \\ 0.2 & 0.7 & 0.1 \\ 0.1 & 0.2 & 0.7 \end{bmatrix}.$$

Prikazati graf ovog sistema. Prikazati trelis ovog sistema. Prikazati vjerovatnoće simbola u stacionarnom stanju.

Rješenje: Na Slici II.10 prikazani su graf tranzicije i trelisa za ovaj Markovljev sistem. U stacionarnom režimu važi:

$$\begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.1 & 0.2 \\ 0.2 & 0.7 & 0.1 \\ 0.1 & 0.2 & 0.7 \end{bmatrix} \begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}.$$

Sistem se može zapisati kao:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.3 & 0.1 & 0.2 \\ 0.2 & -0.3 & 0.1 \\ 0.1 & 0.2 & -0.3 \end{bmatrix} \begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}.$$

Dobijeni homogeni sistem jednačina ne može nam dati odgovarajući rezultat, već moramo da jednu od tri jednačine zamijenimo ograničenjem da suma vjerovatnoća mora biti jednaka 1:

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.3 & 0.1 & 0.2 \\ 0.2 & -0.3 & 0.1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}.$$

Izvršimo sada gausovsku eliminaciju:

$$-0.3p'(a) + 0.1p'(b) + 0.2p'(c) = 0 \Rightarrow p'(b) = 3p'(a) - 2p'(c)$$

$$0.2p'(a) - 0.3p'(b) + 0.1p'(c) = 0 \Rightarrow 0.2p'(a) - 0.9p'(a) + 0.6p'(c) + 0.1p'(c) = 0$$

$$-0.7p'(a) + 0.7p'(c) = 0 \Rightarrow p'(a) = p'(c).$$

Dalje slijedi $p'(b) = p'(a)$. Konačno slijedi da je:

$$p'(a) = p'(b) = p'(c) = \frac{1}{3}.$$

2.15. Poznato je da se entropije $H(X)$, $H(Y)$ i $H(Y|X)$ nalaze u granicama [2,3] bita, [2.5,3.2] bita i [1,1.6] bita, respektivno. U kojim granicama se nalaze združena entropija $H(X,Y)$, te uslovna entropija $H(X|Y)$ i međusobna informacija $I(X;Y)$? Suziti opsege što je više moguće.

Rješenje: Imamo nekoliko relacija koje nam mogu pomoći u određivanju datih ograničenja:

$$I(X;Y) = H(Y) - H(Y|X)$$

$$I(X;Y) = H(X) - H(X|Y)$$

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

$$I(X;Y) = H(X) + H(Y) - H(X,Y).$$

Sve uvedene veličine su nenegativne. Iz druge relacije slijedi da se međusobna informacija nalazi u granicama:

$$I(X;Y) \in [0.9,2.2].$$

Iz druge relacije sada slijedi da je uslovna entropija $H(X|Y)$ u granicama:

$$H(X|Y) = H(X) - I(X;Y) \in [-0.2,2.1].$$

Uz nenegativnost, slijedi da $H(X|Y) \in [0,2.1]$. Iz treće relacije sada slijedi da:

$$H(X,Y) \in [3,4.6] \qquad H(X,Y) \in [2.5,5.3].$$

Imajući na umu usvojena ograničenja, slijedi:

$$H(X,Y) \in [3,4.6],$$

pa ga pokušajmo dalje suziti:

$$H(X,Y) = H(X) + H(Y) - I(X;Y) \in [3.3,5.3].$$

Dakle, suzili smo združenu entropiju na:

$$H(X,Y) \in [3.3,4.6].$$

Proces nije završen jer moramo da nastavimo sa sužavanjem, počevši ponovo od prve relacije. Tek drugi dio treće relacije koristimo da bismo suzili neki od intervala. Pokušali smo, bez uspjeha, da suzimo interval širine 1.9 bita za združenu entropiju. Međutim, uspjeli smo da suzimo interval za uslovnu entropiju $H(X|Y)$, primjenom druge relacije na:

$$H(X|Y) = H(X) - I(X;Y) \in [0.1,2.1].$$

Nastavljamo s posljednjom relacijom, u kojoj su širine intervala za međusobnu informaciju i združenu entropiju po 1.3 bita, pa ih pokušajmo suziti; ali se ni to ne može postići.

Ponovo prolazimo kroz novi krug. Sada probajmo sa sljedećim ograničenjem:

$$H(X) = H(X,Y) - H(Y|X) \in [2.3,3.6].$$

Ovim sužavamo granice sa $H(X)$ na $H(X) \in [2.3,3]$. Za vježbu, pokušajte da izvršite dalje preciziranje granica interval za entropije

2.16. Posmatrajte binarni simetrični kanal, gdje je vjerovatnoća pojavljivanja 1 jednaka p i vjerovatnoća greške pri prenosu simbola kroz kanal jednaka P. Kolika je entropija sistema na ulazu, a kolika na izlazu? Kolika je međusobna informacija stanja na ulazu i izlazu? Kolike su odgovarajuće uslovne entropije?

Rješenje: Ovaj zadatak je uvod za Poglavlje IV, u kojem ćemo više govoriti o komunikacionom kanalu, ali već na ovom primjeru možemo da protumačimo neke od najvažnijih veličina (entropija) potrebnih za opis komunikacionog kanala. Neka su vjerovatnoće bita na ulazu (po uslovima zadatka) jednake p i $1-p$, tako da dobijamo da je entropija na ulazu:

$$H(X) = -p\log_2 p - (1-p)\log_2(1-p) = H(p).$$

Jedinica se dešava na izlazu kada je jedinica poslata s ulaza i ostala ista i kada je s ulaza poslata nula i bila izmijenjena u kanalu:

$$p(Y=1) = p(1-P) + (1-p)P = p + P - 2pP.$$

Slično se određuje vjerovatnoća da je $Y=0$:

$$p(Y=1) = pP + (1-p)(1-P) = 1 - p - P + 2pP.$$

Dakle, entropija skupa Y je:

$$\begin{aligned} H(Y) &= -(p+P-2pP)\log_2(p+P-2pP) - (1-p-P+2pP)\log_2(1-p-P+2pP) \\ &= H(\boldsymbol{p}+P-2pP). \end{aligned}$$

Posmatrajmo sada entropiju $H(Y|X)$. Ako znamo da je poslato $X=0$, onda s vjerovatnoćom P primamo $Y=1$, odnosno s vjerovatnoćom $1-P$ primamo $Y=0$. Dakle, entropija $H(Y|X=0)$ jednaka je entropiji binarnog događaja s vjerovatnoćom P:

$$H(Y|X=0) = -P\log_2 P - (1-P)\log_2(1-P) = H(P).$$

Na isti način važi:

$$H(Y|X=1) = -P\log_2 P - (1-P)\log_2(1-P) = H(P),$$

tako da je uslovna entropija:

$$H(Y\mid X) = \sum_i p(x_i)H(Y\mid X=x_i) = pH(P) + (1-p)H(P) = H(P).$$

Kod združene entropije imamo četiri događaja $\{(X=0,Y=0\}, (X=0, Y=1), (X=1, Y=0), \{X=1, Y=1\})$ sa vjerovatnoćama $\{(1-p)(1-P), (1-p)P, pP, p(1-P)\}$, respektivno, pa je združena entropija:

$$\begin{aligned} H(X,Y) = &-(1-p)(1-P)\log_2(1-p)(1-P) - (1-p)P\log_2(1-p)P - \\ &- pP\log_2 p\boldsymbol{P} - p(1-P)\log_2 p(1-P) = \\ &= -(1-p)(1-P)\log_2(1-p) - (1-p)(1-P)\log_2(1-P) \\ &- (1-p)P\log_2(1-p) - (1-p)P\log_2 P - \\ &- pP\log_2 p - pP\log_2 P - p(1-P)\log_2 p - p(1-P)\log_2(1-P) = \end{aligned}$$

$$= -\log_2(1-p)[(1-p)(1-P)+(1-p)P] - \log_2 p[pP + p(1-P)]$$
$$-\log_2(1-P)[(1-p)(1-P)+p(1-P)] - \log_2 P[(1-p)P + pP]$$
$$= -(1-p)\log_2(1-p) - p\log_2 p - (1-P)\log_2(1-P) - P\log_2 P =$$
$$= H(p) + H(P).$$

Ova relacija je zapravo jednostavno i slijedila iz:

$$H(X,Y) = H(X) + H(Y \mid X).$$

Slično možemo da dobijemo uslovnu entropiju $H(X|Y)$, kao:

$$H(X \mid Y) = H(X,Y) - H(Y) = H(p) + H(P) - H(p + P - 2pP).$$

Konačno, međusobna informacija je:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = H(p) + H(p + P - 2pP) - H(p) - H(P) =$$
$$= H(p + P - 2pP) - H(P).$$

2.17. Dat je Markovljev sistem I reda s alfabetom $X = \{a, b, c, d\}$ i uslovnim vjerovatnoćama datim tabelom:

$$\mathbf{P} = \begin{bmatrix} 0.4 & 0.3 & 0.2 & 0.1 \\ 0.1 & 0.3 & 0.4 & 0.2 \\ 0.24 & 0.36 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.6 & 0.2 \end{bmatrix}.$$

Prikazati graf i trelis ovog sistema. Odrediti vjerovatnoće simbola u stacionarnom stanju.

Rješenje: Sistem jednačina kojim možemo odrediti stacionarne vjerovatnoće možemo zapisati kao:

$$\begin{bmatrix} -0.6 & 0.3 & 0.2 & 0.1 \\ 0.1 & -0.7 & 0.4 & 0.2 \\ 0.24 & 0.36 & -0.8 & 0.2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} p(a) \\ p(b) \\ p(c) \\ p(d) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Iz prve jednačine slijedi:

$$-0.6p(a) + 0.3p(b) + 0.2p(c) + 0.1p(d) = 0 \Rightarrow p(d) = 6p(a) - 3p(b) - 2p(c).$$

Uvrštavanjem u drugu jednačinu, dobijamo:

$$0.1p(a)-0.7p(b)+0.4p(c)+0.2[6p(a)-3p(b)-2p(c)]=0$$
$$1.3p(a)-1.3p(b)=0 \Rightarrow p(a)=p(b)$$
$$p(d)=3p(b)-2p(c).$$

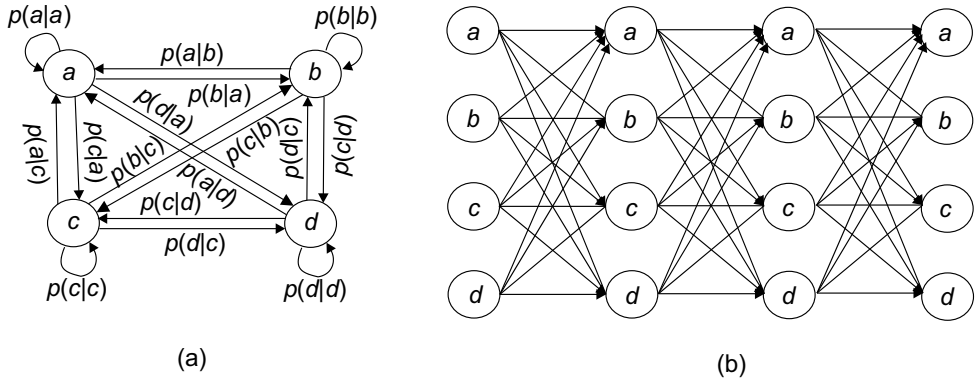
Sada možemo dobijene izraze uvrstiti u treću relaciju:

$$0.24p(b)+0.36p(b)-0.8p(c)+0.2[3p(b)-2p(c)]=0$$
$$1.2p(b)-1.2p(c)=0$$
$$p(b)=p(c) \quad p(a)=p(c) \quad p(d)=p(c).$$

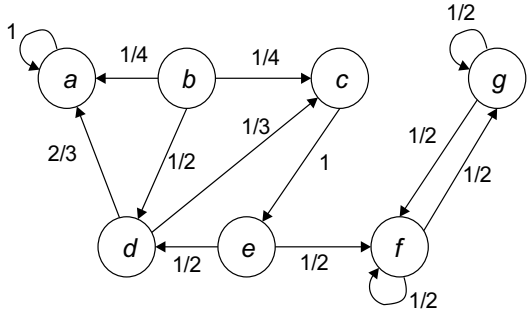
Dakle, kombinujući s posljednjom jednačinom, u stacionarnom režimu dobijamo $p(a)=p(b)=p(c)=\mathrm{p}(d)=1/4$.

Graf tranzicije i trelis su prikazani na Slici II.11.

2.18. Dat je dijagram stanja Markovljevog procesa na Slici II.12. Da li je proces ergodičan? Odrediti tranzijentna, rekurentna, periodična, aperiodična i apsorbujuća stanja u dijagramu. Prikazati matricu tranzicije sistema.



Slika II.11. Kvaternarni alfabet kao Markovljev sistem I reda: (a) Graf tranzicije; (b) Trelis



Slika II.12. Markovljev sistem prvog reda za zadatak 2.18

Rješenje: Stanja *b*, *c*, *d* i *e* su tranzijentna jer se iz barem jednog stanja u koje se iz njih može doći ne možemo vratiti u to stanje. Rekurentna (esencijalna) stanja su *a*, *f* i *g* jer se može izvršiti povratak iz svakog stanja u koje se iz njih može doći. Stanja *c*, *d*, *e* su periodična jer se možemo kretati između njih periodičnom putanjom dužine $p = 3$ $p > 1$). Stanje *a* je apsorbujuće, jer kada se stigne u ovo stanje, ono se više ne može napustiti. Predmetni proces nije ergodičan, jer se iz stanja *a* ne može u konačnom broju koraka stići u sva druga stanja. Kao što smo rekli, u pitanju je apsorbujuće stanje.

Matrica tranzicije za ovaj sistem je:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2/3 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \end{bmatrix}.$$

Studenti mogu, između ostalog, za samostalan rad da sračunaju entropije za sve sisteme koji su razmatrani u okviru riješenih zadataka u Poglavlju I.

II.3.2 Softverska realizacija

Ponovo napominjemo da nam nije za cilj da govorimo o nekim specijalnim funkcijama niti da detaljno pregledamo opcije vezane za MATLAB realizaciju pojednih koncepata, već samo da rudimentarno damo neke od osnovnih elemenata softverske realizacije.

A. Pretpostavimo da nam je dat vektor vjerovatnoća $p = [0.3\ 0.2\ 0.1\ 0.1\ 0.08\ 0.07\ 0.07\ 0.06\ 0.02]$. Odrediti entropiju predmetnog alfabeta:

```
>> p=[0.3 0.2 0.1 0.1 0.08 0.07 0.07 0.06 0.02]
>> sum(-p.*log2(p))          %entropija u bitima
   2.834890334005890
>> sum(-p.*log10(p))         %entropija u ditima
   0.853387024953655
>> sum(-p.*log(p))           %entropija u natovima
   1.964996242212825
>> sum(-p.*log2(p)/log2(4))  %entropija računata za alfabet sa osnovom 4
   1.417445167002945
```

Vidimo da se entropije s osnovama 2, 10 i *e* dobijaju korišćenjem ugrađenih funkcija za logaritam, dok se u slučaju potrebe da se logaritam računa s nekom drugom bazom radi putem log(*x*)/log(*b*), gdje je *x* vrijednost za koju se logaritam računa, dok je *b* alternativna baza.

B. Odrediti entropije za događaje date u Primjeru I.2.

Prvo formiramo matricu združenih vjerovatnoća:

```
>> Pxy=zeros(11,6);
>> Pxy(1,1)=1/36; Pxy(3,2)=1/36;Pxy(5,3)=1/36;
>> Pxy(7,4)=1/36;Pxy(9,5)=1/36;Pxy(11,6)=1/36;
>> Pxy(2,2)=2/36;Pxy(3:4,3)=2/36;Pxy(4:6,4)=2/36;
>> Pxy(5:8,5)=2/36;Pxy(6:10,6)=2/36
```

Direktno računanje entropije ne daje rezultat zbog logaritma od nule.

```
>> -sum(Pxy(:).*log2(Pxy(:)))
   NaN
```

Stoga na sve vjerovatnoće dodajemo eps (vrijednost koja je mali pozitivni broj $eps=2^{-52}$):

```
>> format long
>> -sum(Pxy(:).*log2(Pxy(:)+eps))
   4.336591668108972
```

Možemo da provjerimo da li ovakvo računanje daje neznatno odstupanje od tačnog rezultata:

```
>> -6*(1/36)*log2(1/36)-15*(2/36)*log2(2/36)
ans =
   4.336591668108978
```

Razlika je stvarno zanemarljiva, što nam kaže da se eps može koristiti u izrazima za entropiju. Sračunajmo marginalne entropije sumiranjem združene entropije po kolonama i vrstama:

```
>> Px=sum(Pxy');
>> Py=sum(Pxy);
```

Dijeljenjem združene entropije sa marginalnim, dobijamo uslovne entropije:

```
>> Pyzx=Pxy./kron(Px',ones(1,6));
>> Pxzy=Pxy./kron(Py,ones(11,1));
```

Naredba kron se koristi za Kronekerovo množenje, odnosno svaki element prve matrice množi čitavu drugu matricu, formirajući matricu većih dimenzija. Ovdje je

upotrijebljeno da bi se izjednačile dimenzije marginalnih i združene vjerovatnoće. Vidimo da su dobijeni odgovarajući rezultati:

```
>> Pxzy
1.0000   0        0        0        0        0
0        0.6667   0        0        0        0
0        0.3333   0.4000   0        0        0
0        0        0.4000   0.2857   0        0
0        0        0.2000   0.2857   0.2222   0
0        0        0        0.2857   0.2222   0.1818
0        0        0        0.1429   0.2222   0.1818
0        0        0        0        0.2222   0.1818
0        0        0        0        0.1111   0.1818
0        0        0        0        0        0.1818
0        0        0        0        0        0.0909
```

Sada se može, bez daljih problema, pristupiti računanju ostalih entropija, marginalnih:

```
>> sum(-Px.*log2(Px))
   3.2744
>> sum(-Py.*log2(Py))
   2.3200
```

i uslovnih:

```
>> sum(-Pxy(:).*log2(Pxzy(:)+eps))
   2.0166
>> sum(-Pxy(:).*log2(Pyzx(:)+eps))
   1.0622
```

Definiciono se združena informacija može dobiti kao:

```
>> sum(sum(Pxy.*log2((Pxy+eps)./kron(Px',Py))))
   1.2578
```

C. Odrediti stacionarne vjerovatnoće za Markovljev sistem I reda iz Primjera II.2.

Matrica tranzicije je:

```
>> P=[1/3 1/3 1/3;1/4 1/2 1/4;1/4 1/4 1/2];
```

Izvršimo njeno transponovanje i oduzmimo jediničnu matricu (funkcija eye u MATLAB-u):

```
>> Pt=P';
>> Pt=Pt-eye(size(Pt));
```

Nakon toga zamijenimo posljednji red vrstom koja ima sve jedinice:

```
>> Pt(end,:)=ones(1,size(P,2));
```

i, konačno, odredimo vjerovatnoće u stacionarnom režimu (uočite da smo za transponovanje matrice realnih brojeva koristili ‘):

```
>> inv(Pt)*[0 0 1]'
  0.2727
  0.3636
  0.3636
```

Razlomke možete dobiti:

```
>> rats(ans)
    3/11
    4/11
    4/11
```

Istina, ova naredba može da produkuje i netačan ili neočekivan rezultat zbog načina kako tretira tačnost u prikazivanju razlomaka.

POGLAVLJE III

KODIRANJE IZVORA

III. KODIRANJE IZVORA

Vidjeli smo da pouzdan i efikasan prenos informacija počiva na dva fundamentalna koncepta: kodiranju izvora i kodiranju kanala. Kodiranje (i dekodiranje) izvora je tema ovog poglavlja. Zadatak kodiranja izvora je da skupi informaciju iz nekog izvora na što je moguće manji memorijski prostor radi manjih memorijskih zahtjeva i manjeg zauzimanja kanala za slanje ovakvih poruka. Postoje dva tipa kodova izvora: **kodovi bez gubitaka** i **kodovi sa gubicima**. Dekodiranje kodova bez gubitaka daje potpuno ispravnu polaznu informaciju, što nije slučaj kod druge klase kodova. Sa druge strane, kodovi sa gubicima daju mogućnost veće **kompresije**, odnosno većeg sažimanja informacije, ali zbog njihove složenosti biće izostavljeni u ovom materijalu. Materija u ovom poglavlju podijeljena je u šest osnovnih cjelina. U prvom dijelu se pokazuje da je kompresija podataka moguća kroz rudimentaran prikaz **asimptotske ekviparticione osobine**. Zatim dajemo osnovne pojmove i koncepte u kompresiji bez gubitaka, praćene matematičkom podlogom kompresije bez gubitaka u vidu nekoliko teorema i stavova. Prije prelaska na same kodove za kodiranje izvora, dajemo nekoliko **pomoćnih kodova** koji se u teoriji informacija i kodova ne koriste samostalno, ali se često kombinuju s drugim postupcima. Zatim uvodimo kodiranje entropije i njenog najboljeg predstavnika − **Hafmenov** (engl. Huffman) **kod** sa nekoliko podvarijanti. Među **kodovima zasnovanim na rječniku**, predstavili smo ponovo najbolju grupu kodova − **LZ** i **LZW kodove** sa nekoliko podvarijanti. Poglavlje zaključujemo s **aritmetičkim kodovima**.

III.1 Asimptotska ekviparticiona osobina

Asimptotska ekviparticiona osobina (često se skraćeno označava **AEP**, kao skraćenica od naziva na engleskom jeziku – *Asymptotic Equipartition Property*) suštinski predstavlja formulaciju odgovarajuće teoreme koja dokazuje da je kompresija

podataka moguća, odnosno u našem slučaju kodiranje izvora, te uvodi osnovne veličine i parametre vezane za kodiranje izvora.

Prije neposredne formulacije ove teoreme, dajemo jedan primjer koji će nam poslužiti da shvatimo osnovne činjenice vezane za ovu bitnu osobinu.

Posmatrajmo binarni alfabet sa vjerovatnoćom pojave 0, koja je jednaka $q = 1 - p = 0.7$ i vjerovatnoćom pojave 1, koja je jednaka $p = 0.3$. Entropija ovog skupa je jednaka:

$$H(p) = H(0.3) = -0.3\log_2 0.3 - 0.7\log_2 0.7 \approx 0.8819 \text{ bita.}$$

Pretpostavimo sada da šaljemo slučajne poruke od $n = 10$ bita tako da su biti nezavisni i identično distribuirani (engl. *i.i.d. – independent and identically distributed*). Ukupna entropija ove poruke je:

$$nH(p) = 8.819 \text{ bita.}$$

Ukupan broj kombinacija koje se mogu na ovaj način generisati je $2^n = 1024$ i za njihovo kodiranje je potrebno $n = 10$ bita. Pretpostavimo da imamo neku partikularnu sekvencu u kojoj se pojavljuje r jedinica. Vjerovatnoća pojavljivanja te partikularne sekvence je: $p^r q^{n-r} = p^r (1-p)^{n-r} = 0.3^r 0.7^{10-r}$. Broj kombinacija n bita u kojima se pojavljuje r jedinica jednak je:

$$\binom{n}{r} = \frac{n!}{r!(n-r)!},$$

tako da je vjerovatnoća svih kombinacija u kojima se pojavljuje r jedinica u n bita jednaka:

$$\binom{n}{r} p^r (1-p)^{n-r} = \frac{10!}{r!(10-r)!} 0.3^r 0.7^{10-r}.$$

U Tabeli III.1 prikazane su vjerovatnoće pojedinačnih događaja sa r jedinica, broj kombinacija i ukupna vjerovatnoća svih sekvenci sa r jedinica.

Uočimo da po vrstama ukupne vjerovatnoće rastu, dok ne dostignu 0.262 za sekvence s tri jedinice i sedam nula, nakon čega opadaju. Matematičko očekivanje broja jedinica u poruci je:

$$np = 3,$$

pa smo dobili da se maksimum ukupne vjerovatnoće poklapa s očekivanim brojem jedinica u poruci. Pošto se radi u i.i.d. procesu, entropija ovog događaja je, kao što smo vidjeli, $nH(p) = 8.819$. Pogledajmo sada veličinu:

$$2^{-nH(p)} \approx 0.02214.$$

r	$0.3^r 0.7^{10-r}$	$10!/[r!(10-r)!]$	**Ukupno**
0	0.0282	1	0.0282
1	0.0121	10	0.121
2	0.005	45	0.225
3	**0.0022**	**120**	**0.264**
4	0.00095	210	0.1995
5	0.0004	252	0.1008
6	0.00017	210	0.036
7	0.000075	120	0.009
8	0.000032	45	0.00144
9	0.0000138	10	0.000138
10	0.0000059	1	0.0000059

Tabela III.1. Broj jedinica u sekvenci, vjerovatnoća pojedinačne sekvence i ukupna vjerovatnoća za dati broj jedinica u sekvenci za korišćeni primjer

Dakle, svaka sekvenca iz skupa onih koje se najčešće pojavljuju (nazivaju se **tipičnim sekvencama**) javlja se s vjerovatnoćom koja je približno $2^{-nH(p)}$. U podsekcijama koje slijede biće formulisana i djelimično dokazana asimptotska ekviparticiona osobina sa formalnom definicijom tipične sekvence. Za većinu čitalaca dokaz je previše teorijska materija, pa je sekcija označena zvjezdicom da bi se naglasilo da se taj dio može i preskočiti u zavisnosti od afiniteta čitaoca.

III.1.1 Formulacija teoreme

Teorema III.1. Pretpostavimo da posmatramo sekvencu od n uzastopnih primljenih simbola nekog alfabeta $(x_1, x_2, ..., x_n)$.

(1) Predmetna sekvenca pripada skupu **tipičnih sekvenci** $(x_1, x_2, ..., x_n) \in A_\varepsilon^{(n)}$ sa vjerovatnoćom:

$$H(X) - \varepsilon \le -\log p(x_1, x_2, ..., x_n)/n \le H(X) + \varepsilon.$$

U predmetnoj formulaciji ε označava proizvoljno malu pozitivnu vrijednost.

(2) Ukupna vjerovatnoća tipične sekvence može se učiniti proizvoljno bliskom 1 $\Pr\left\{A_\varepsilon^{(n)}\right\} > 1 - \varepsilon$ za n dovoljno veliko.

(3) Broj elemenata u tipičnom skupu (koristićemo oznake || || za broj elemenata u skupu) nalazi se u granicama: $(1-\varepsilon)2^{n(H(X)+\varepsilon)} \le \left\|A_\varepsilon^{(n)}\right\| \le 2^{n(H(X)+\varepsilon)}$.

*III.1.2 Dokaz teoreme**

Prije dokazivanja najbitnijeg stava predmetne teoreme, dajemo definiciju konvergencije po vjerovatnoći i neke stavove koji se koriste u dokazivanju same teoreme.

Definicija III.1. Niz (red) $X_1, X_2, ..., X_n$ konvergira po vjerovatnoći nekoj vrijednosti X ako je $\Pr[|X_n - X| < \varepsilon] > 1 - \delta$ za svako $n > n_0$, gdje su ε i δ proizvoljne pozitivne konstante.

Primjer III.1. Neka je X_i niz slučajnih promjenljivih i neka je:

$$S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

srednja vrijednost ovog niza za prvih n elemenata. Neka je srednja vrijednost svakog pojedinačnog elementa iz ovoga skupa S i neka je varijansa slučajnog procesa (šuma) koji je zahvatio našu opservaciju σ^2. Tada elemente niza X_i možemo zapisati kao:

$$X_i = S + \nu_i .$$

Neka su ν_i nezavisni i na isti način distribuirani (i.i.d.), odnosno da važi:

$$\mathrm{E}\{\nu_i\} = 0$$

$$\mathrm{E}\{\nu_i^2\} = \sigma^2$$

$$\mathrm{E}\{\nu_i \nu_j\} = 0 \quad \text{za} \quad i \neq j .$$

Tada je srednja vrijednosti od S_n jednaka:

$$\mathrm{E}\{S_n\} = \frac{1}{n}\sum_{i=1}^{n}[S + E\{\nu_i\}] = S + \frac{1}{n}\sum_{i=1}^{n}\mathrm{E}\{\nu_i\} = S,$$

dok je varijansa:

$$\mathrm{E}\{[S_n - S]^2\} = \mathrm{E}\left\{\left[\frac{1}{n}\sum_{i=1}^{n}[S + \nu_i] - S\right]\right\} = \mathrm{E}\left\{\left[\frac{1}{n}\sum_{i=1}^{n}\nu_i\right]^2\right\} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{E}\{\nu_i \nu_j\} =$$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma^2\delta(i-j) = \frac{\sigma^2}{n}.$$

Dakle, varijansa ovog procesa opada ka nuli kako n raste, pa se može učiniti proizvoljno malom. Stoga možemo zaključiti da se i vjerovatnoća da $|S_n - S|$ uzima proizvoljno malu vrijednost može učiniti proizvoljno bliskom 1 za neko dovoljno veliko n.

Napomena*. Postoji mogućnost da se predmetna konvergencija preciznije formuliše, te da se kvantifikuje brzina približavanja po vjerovatnoći (brzina konvergencije po vjerovatnoći u smislu Markovljeve relacije) koja za pozitivnu slučajnu promjenljivu i $\delta > 0$ zadovoljava:

$$\Pr\left[X \geq \varepsilon\right] \leq E\left[X\right] / \delta.$$

Odavde se može zapisati Čebiševljeva nejednakost (rus. Чебышёв) za slučajnu promjenljivu Y sa srednjom vrijednošću μ i varijansom μ

$$\Pr\left[\left|Y - \mu\right| > \varepsilon\right] \leq \sigma^2 / \varepsilon^2. \square$$

Za zainteresovane preporučujemo adekvatnu literaturu iz matematičke statistike.

Sada se možemo vratiti dokazu (pojedinih stavova) naše teoreme.

Teorema III.2. Neka su $X_1, X_2, ..., X_n$ i.i.d. slučajne promjenljive sa funkcijom gustine raspodjele $p(x)$. Tada $-\frac{1}{n}\log p\left(X_1, X_2, ..., X_n\right)$ konvergira po vjerovatnoći ka entropiji $H(X)$.

Dokaz. Za nezavisne i na isti način distribuirane slučajne promjenljive važi:

$$-\frac{1}{n}\log p\left(X_1, X_2, ..., X_n\right) = -\frac{1}{n}\sum_{i=1}^{n}\log p\left(X_i\right).$$

Kako su X_i nezavisne slučajne promjenljive s istom gustinom raspodjele, isto važi i za $\log p(X_i)$. Na osnovu prethodnog primjera, srednja vrijednost odbiraka (u ovom slučaju $\log p(X_i)$) konvergira ka srednjoj vrijednosti ansambla. Srednja vrijednost ansambla je jednaka:

$$-\sum_{i=1}^{n} p(X_i)\log p\left(X_i\right) = H(X).$$

Dakle, zaključujemo da $-\frac{1}{n}\log p\left(X_1, X_2, ..., X_n\right)$ konvergira po vjerovatnoći ka entropiji $H(X)$, čime smo zaključili predmetni dokaz.□

Ako se sada vratimo na Teoremu III.1, slijedi da je $\log p(x_1, x_2, ..., x_n)$ približno $-nH(X)$, odnosno da je:

$$p(x_1, x_2, ..., x_n) \approx 2^{-nH(X)}.$$

Ovim smo suštinski dokazali prvi stav Teoreme III.1. Stav (2) ove teoreme slijedi iz definicije konvergencije po vjerovatnoći. Vjerovatnoća da sekvenca pripada tipičnom skupu može se zapisati na sljedeći način:

$$\left| -\frac{1}{n}\log p\left(X_1, X_2, ..., X_n\right) - H(X) \right| \leq \varepsilon,$$

što se može za dovoljno veliko n učiniti proizvoljno bliskim jedan.

III.1.3 Tumačenje teoreme

Za većinu čitalaca sam dokaz teoreme (čiji je dio dat u Poglavlju III.1.2) nije od prevelikog značaja, već je mnogo bitnije da se razumiju njene posljedice. Prva činjenica do koje smo došli je da je vjerovatnoća pojave sekvence iz tipičnog skupa:

$$p(x_1, x_2, ..., x_n) \approx 2^{-nH(X)},$$

što smo u našem primjeru i vidjeli, odnosno kod skupa sa samo 10 elemenata dostigli smo da se $p(x_1, x_2, ..., x_n)$ i $2^{-nH(X)}$ razlikuju na petoj decimali, odnosno za manje od 0.8%. Može se, naravno, postaviti i razumno pitanje: Šta se dešava kada događaji (simboli) x_i nisu nezavisni? Kod gotovo svih poruka postoji veza između uzastopnih simbola. Odgovor je jednostavan: vjerovatnoća tipične sekvence, u tom slučaju $p(x_1, x_2, ..., x_n)$, približno je jednaka $2^{-H(x_1, x_2, ..., x_n)}$, gdje je sada u eksponentu združena entropija n simbola (za i.i.d. procese iznosi $nH(X)$). Drugi stav je znatno manje očigledan. Tvrdimo da je ukupna vjerovatnoća tipične sekvence (ili okolina) ove pozicije približno 1. U našem primjeru dobijena je vjerovatnoća od samo 0.264, pa čak i kada objedinimo dva okolna događaja (sekvence sa dvije jedinice s vjerovatnoćom 0.225 i sekvence sa četiri jedinice, koje se pojavljuju s ukupnom vjerovatnoćom 0.1995) dobijeni rezultat je nešto ispod 0.69. Međutim, riječ je samo o ilustrativnom primjeru, dok je situacija za sisteme sa većim brojem n povoljnija. Na primjer, za $n = 1000$ ponovili smo eksperiment u generisanju slučajne sekvence 100000 puta. Ni u jednom slučaju nismo dobili sekvencu sa manje od 240 jedinica niti sa više od 370 jedinica, dok je u granicama od 280 do 320 jedinica bilo generisano preko 85% svih sekvenci! Dakle, ovim jednostavnim eksperimentom mogli smo da pokažemo važenje tvrdnje da kako dužina sekvence raste, povećava se broj elemenata koji pripada tipičnoj sekvenci i to na način da skoro svaka generisana sekvenca pripada ovom skupu.

III.1.4 Primjena teoreme

Posmatrajmo sljedeći slučaj iz našeg primjera. Tipični skup ima 120 elemenata. Ovih 120 elemenata možemo da kodiramo sa 7 bita ($2^7 = 128 > 120$). Da bismo razlikovali ovakav skup od ostatka ovog sistema, dodajmo svakoj sekvenci, na njenom početku, početni bit (prefiks) 1, kao naznaku da je u pitanju „prekodirana" tipična sekvenca. Dakle, tipične sekvence kodiramo sa ukupno $1 + 8$ bita. Ostale sekvence moramo da kodiramo prefiksnom 0, čime smo povećali dužinu koda za ove sekvence na $10 + 1 = 11$ bita. Prefiksna nula naglašava da je u pitanju sekvenca

van tipičnog skupa. Sada imamo prvi kod za kodiranje izvora sa prosječnom dužinom kodne riječi:

$$0.264 \times 8 + 0.736 \times 11 = 10.208 \text{ bita/sekvenci.}$$

Vidimo da predmetnim postupkom nismo ništa dobili, već smo pogoršali naše kodiranje, odnosno da nam za naših 10 bita u prosjeku sada treba 0.208 bita/sekvenci više. Da li se ovo može popraviti? Na svu sreću, ovo je moguće popraviti čak i u ovim veoma primitivnim uslovima. Pošto sa 7 bita možemo kodirati 128 poruka, a mi smo kodirali samo 120, dodajmo ovom skupu dodatnih 8 poruka. Neka to bude najvjerovatnija od svih poruka sa svim nulama (vjerovatnoća 0.0282) i sedam od deset poruka sa po jednom jedinicom (vjerovatnoća $7 \times 0.0121 = 0.0847$). Sada 128 poruka, koje smo selektovali da kodiramo sa 7 bita, ima vjerovatnoću od ukupno:

$$0.264 + 0.0282 + 0.0847 \approx 0.3769.$$

Kodiranje s opisanim prefiksnim postupkom daje **prosječnu dužinu kodne riječi** (pojam koji ćemo kasnije preciznije definisati):

$$0.3769 \times 8 + 0.6231 \times 11 \approx 9.8693 \text{ bita/sekvenci.}$$

Dakle, čak i predmetni, prilično primitivan postupak daje smanjivanje prosječne dužine kodne riječi. Prije nego pokažemo da ovaj postupak može još da se poboljša, moramo da uvedemo još jedan pojam.

Definicija III.2. Skup sekvenci koje se mogu formirati po nekom pravilu sa n slučajnih promjenljivih, takvih da $\Pr\left(B_\delta^{(n)}\right) \geq 1 - \delta$, gdje je δ neka zadata vjerovatnoća, naziva se **visokovjerovatni skup**.

Visokovjerovatni skup je, po definiciji, veoma sličan tipičnoj sekvenci, ali bez restrikcije vezane za način formiranja ovog skupa (kod tipične sekvence imali smo određene restrikcije kako se ova sekvenca formira). Može se pokazati (teoremom koja ima netrivijalnu formulaciju i dokaz) da je broj elemenata u visokovjerovatnom skupu istog reda veličine kao kod tipične sekvence. Što je ovdje još značajnije (takođe neće biti dokazivano), svaki visokovjerovatni skup ima veliki broj zajedničkih elemenata sa tipičnim skupom. Još preciznije: nije moguće formirati visokovjerovatni skup a da nema veliki broj zajedničkih elemenata s tipičnim skupom.

Posmatrajmo ponovo naš primjer sa kratkim kodnim riječima dužine $n = 10$. Uzećemo da se u visokovjerovatnom skupu sa 128 elemenata nalazi 128 najvjerovatnijih sekvenci: jedna sa svim nulama (vjerovatnoća 0.0282), deset sa jednom jedicinom (ukupna vjerovatnoća 0.121), 45 sa dvije jedinice (ukupna vjerovatnoća 0.225) i preostalih 72 sa tri jedinice iz tipičnog skupa (ukupna vjerovatnoća $72 \times 0.0022 = 0.1584$). Dakle, ovaj visokovjerovatni skup ima presjek sa skupom tipičnih sekvenci od 72 elementa ($72/128 \approx 56.25\%$). Ukupna vjerovatnoća simbola

ovog visokovjerovatnog skupa je: ≈ 0.5326. Ako primijenimo prethodno opisani način kodiranja prefiksnim bitom, dobijamo da je prosječna dužina kodne riječi:

$$0.5326 \times 8 + 0.4674 \times 11 = 9.4022 \text{ bita/sekvenci},$$

čime smo postigli dodatnu uštedu u kodiranju (koja kasnije za posljedicu ima smanjene memorijske zahtjeve i efikasnije komuniciranje uz manje troškove).

III.1.5 AEP za zavisne događaje

Asimptotsku ekviparticionu osobinu smo dokazali za događaje koji su nezavisni i na isti način distribuirani. Postavlja se pitanje: šta se dešava kod praktičnih slučajeva kada događaji (ili simboli prilikom slanja informacija) nisu ovakvi? Bez ulaženja u detaljno razmatranje ove problematike, ne postoji prevelika razlika u odnosu na slučaj koji smo prethodno razmatrali. U slučaju nezavisnih i na isti način distribuiranih događaja, entropija sekvence jednaka je sumi n entropija pojedinačnog događaja:

$$H(x_1, x_2, ..., x_n) = nH(x),$$

dok u slučaju zavisnih i/ili događaja gdje se distribucija simbola mijenja, ovo pojednostavljenje ne važi. Stoga, u iskazu teoreme očekujemo da je vjerovatnoća tipične sekvence približno:

$$p(x_1, x_2, ..., x_n) \approx 2^{-H(x_1, x_2, ..., x_n)}.$$

Na sličan način sada se može zaključiti da je broj elemenata u skupu ovakvih tipičnih sekvenci približno $2^{H(x_1, x_2, ..., x_n)}$. Zainteresovani za dalje izučavanje ove materije mogu da konsultuju neke od knjiga datih u literaturi na kraju ove knjige.

III.2 Kompresija bez gubitaka – pojmovi i teoreme

Osnovni smisao asimptotske ekviparticione osobine je da pokaže mogućnost kompresije podataka. Vidjeli smo da se tipična sekvenca pojavljuje većinom vremena za veliko n. Znamo da ASCII kod daje istu dužinu podatka za svaki od elemenata skupa. Međutim, vidjeli smo da bismo ostvarili kompresiju ako bismo elemente (sekvence) koji se često pojavljuju zamijenili manjim brojem bita, a one koji se pojavljuju veoma rijetko zamijenili velikim brojem bita. Mi smo to postigli korišćenjem koncepta tipične sekvence. Naime, vidjeli smo da elemenata u tipičnoj sekvenci ima približno $2^{nH(X)}$. Svaka od ovih sekvenci može se kodirati sa približno $nH(X)+1$ bit, uključujući prefiksni bit. Svaka sekvenca koja nije u tipičnom skupu može se kodirati sa $n+1$ bit. Čak i ovako gruba šema dala nam je mogućnost ostvarivanja određene kompresije. Sada idemo korak dalje, da vidimo kolike su stvarne mogućnosti ostvarivanja kompresije bez gubitaka.

Prije nego pokažemo fundamentalne rezultate vezane za mogućnost ostvarivanja optimalne kompresije bez gubitaka, moramo da uvedemo i da razumijemo određeni skup pojmova.

Definicija III.3. Kod se naziva **singularnim** ako se dva simbola ulaznog alfabeta preslikavaju u istu kodnu riječ. Ovakvi kodovi se ne mogu jednoznačno dekodirati (preslikati iz skupa kodnih simbola u polazni alfabet), odnosno ovakvi kodovi ne mogu da obezbijede kompresiju bez gubitaka.

Primjer III.2 Pretpostavimo da imamo alfabet sa četiri simbola $\{a, b, c, d\}$. Kod prikazan u Tabeli III.2 je singularan jer su kodovi za kodne simbole b i d isti i ne mogu se razlučiti u procesu dekodiranja.

Posmatrajmo slučaj da smo umjesto singularnog koda primijenili kod iz Tabele III.3, koji očigledno nije singularan i koji je dobijen promjenom koda za simbol 'd' iz prethodne tabele. Postavlja se pitanje: Da li je ovakav kod upotrebljiv? Odgovor na ovo pitanje je: „Ne". Naime, mi u kanal šaljemo serije simbola, a ne pojedinačne simbole i naša je težnja da se te serije simbola dekodiraju, a u ovom slučaju to očigledno nije moguće jer je kombinacija 'aa' kodirana sa 00, čime je kodirano i samo 'd', pa ne postoji ni mogućnost da se ova dva slučaja mogu razdvojiti. Dakle, da bi kod bio upotrebljiv, za svaku sekvencu simbola **ulaznog alfabeta**

$$x_1 x_2 \cdots x_Q \cdots$$

mora da generiše sekvencu simbola kodnih riječi (u nekom **izlaznom alfabetu**)

$$C(x_1)C(x_2)\cdots C(x_Q)\cdots$$

(gdje smo sa $C()$ označili pravilo preslikavanja iz ulaznog alfabeta u kod) tako da se ulazna sekvenca može nedvosmisleno dekodirati. Ovakvi kodovi se nazivaju **jednoznačno dekodabilnim** i jedino su nam oni od interesa. Posmatrajmo sada Tabelu III.4 sa tri koda kojima kodiramo iste simbole ulaznog alfabeta. Kod 1 je očigledno jednoznačno dekodabilan. To je, zapravo, kôd kod kojega svaki kodni simbol ulaznog alfabeta kodiramo sa različitom riječju simbola izlaznog alfabeta konstantne dužine. Kod 2 je takođe jednoznačno dekodabilan! Pretpostavimo da

X	a	B	c	d
Kod	0	01	11	01

Tabela III.2. Primjer singularnog koda

X	a	B	c	d
Kod	0	01	11	00

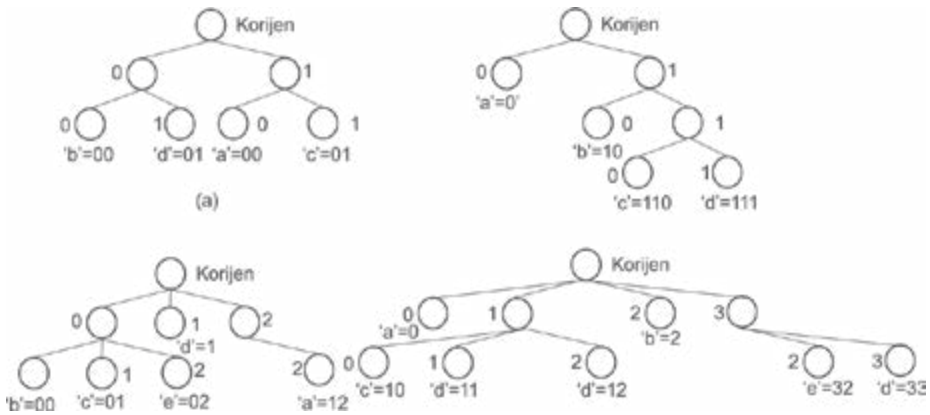
Tabela III.3. Primjer nesingularnog koda koji nije pogodan za kompresiju bez gubitaka (kodiranje izvora)

X	*a*	*B*	*c*	*d*
Kod 1	10	00	11	01
Kod 2	10	00	11	110
Kod 3	0	10	110	111

Tabela III.4. Primjeri tri jednoznačno dekodabilna koda

smo u kanal poslali sekvencu ‘ccba’, ona je kodirana kao 11110010. Kada primimo prvu jedinicu, mi nismo sigurni koji smo kodni simbol primili. Kada primimo drugu jedinicu i dalje nismo sigurni što smo primili, jer to može biti dio od *c*, ali i dio od *d*. Kada primimo treću jedinicu, sigurni smo da su prve dvije predstavljale simbol *c*. Primamo četvrtu jedinicu i ponovo ne znamo koji simbol smo primili. Nakon što primimo nulu, naše neznanje nije smanjeno, jer kombinacija 110 može biti *d*, ali može biti i dio od *cb*. Kada primimo narednu nulu, sigurni smo da smo primili *cb*, i na kraju stiže 10 koje dekodiramo kao *a*. Premda je kod jednoznačno dekodabilan, on je za nas složen jer ne možemo, nakon primljenih bita, da odgonetnemo koji je simbol prenesen. Zaključujemo da je, pored jednoznačne dekodabilnosti, bitan faktor i mogućnost jednostavne odluke o poslatim simbolima.

Kod 3 u Tabeli III.4 ponovo je jednoznačno dekodabilan, a kao prednost u odnosu na kôd 2 kod njega se (kao i kod koda 1) odluka može donijeti sa posljednjim primljenim bitom poslatog simbola. Kada god primimo simbol nula, znamo šta smo primili i, na osnovu toga koliko je jedinica bilo u odnosu na prethodnu odluku, konstatujemo primljeni simbol polaznog alfabeta: ako nema jedinica, već samo nula, primili smo *a*, ako je bila jedna jedinica, tumačimo primljeno 10 kao *b*, dok ako su bile dvije jedinice, tumačimo primljeno 110 kao *c*. Ovakvi kodovi se često nazivaju **koma kodovi** jer u ovom slučaju nula služi kao zarez (koma), nakon kojeg primamo novi simbol. Ako smo primili tri uzastopne jedinice, konstatujemo da



Slika III.1. Prefiksni (trenutni kodovi): (a) Binarni kod 1 iz Tabele III.4; (b) Binarni kod 3 iz Tabele III.4; (c) Primjer ternarnog koda; (d) Primjer kvaternarnog koda

je primljen simbol d ulaznog alfabeta. Kodovi kod kojih se odluka o primljenom simbolu donosi sa posljednjim bitom (ili simbolom izlaznog alfabeta) nazivaju se **trenutnim** (odluka se donosi trenutno) ili **prefiksnim**. Termin **prefiksni** potiče iz činjenice da nijedan prefiks kodne riječi nije kodna riječ, odnosno da podskup simbola kodne riječi s početka kodne riječi ne predstavlja kodnu riječ. Naša je želja da su nam kodovi prefiksni, jer su onda jednostavni za dekodiranje. Prefiksni kodovi se vrlo jednostavno vizuelizuju primjenom (binarnog) drveta (Slika III.1). Na slici su, pored kodova 1 i 3 iz Tabele III.4, prikazana dva nebinarna koda: **ternarni**, sa tri simbola izlaznog alfabeta, i **kvaternarni**, sa četiri simbola izlaznog alfabeta. U daljem tekstu ćemo broj simbola izlaznog alfabeta označiti sa D. Ono što se odmah može zaključiti jeste to da kod trenutnih (prefiksnih) kodova listovi drveta predstavljaju kodne riječi, te da nema kodne riječi koja nije na listu (koja je prefiks neke druge kodne riječi).

U Tabeli III.4 kodovi 1 i 3 ispunjavaju osobine jednoznačne dekodabilnosti i trenutnosti. Koji je od ova dva koda bolji?

Odgovor na predmetno pitanje ne može se dati bez utvrđivanja kriterijuma po kojem će se procjenjivati kvalitet kodova. Taj parametar je **prosječna dužina kodne riječi**. Prosječna dužina kodne riječi se definiše kao:

$$\bar{L} = \sum_{i=1}^{N} l(x_i)\, p(x_i),$$

gdje je N broj simbola (kardinalnost) alfabeta, x_i su simboli (ulaznog) alfabeta, $p(x_i)$ je vjerovatnoća simbola ulaznog alfabeta, dok je $l(x_i)$ dužina kodne riječi izlaznog alfabeta kojom je kodirano x_i.

Primjer III.3. Posmatrajmo sljedeće dvije situacije:

(a) vjerovatnoće simbola ulaznog alfabeta su $p(\text{‘}a\text{’}) = p(\text{‘}b\text{’}) = p(\text{‘}c\text{’}) = p(\text{‘}d\text{’}) = 1/4$;

(b) vjerovatnoće simbola ulaznog alfabeta su $p(\text{‘}a\text{’}) = 1/2$, $p(\text{‘}b\text{’}) = 1/4$, $p(\text{‘}c\text{’}) = p(\text{‘}d\text{’}) = 1/8$.

Odredite koji je od dva trenutna, jednoznačno dekodabilna koda iz Tabele III.4 bolji za kodiranje u ove dvije situacije.

Rješenje: Prosječna dužina kodne riječi za kod 1 je u oba slučaja jednaka $\bar{L} = 2$ bita / simbolu. Za kod 3 je, u prvoj situaciji, prosječna dužina kodne riječi:

$$\bar{L} = 1 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{4} + 3 \cdot \frac{1}{4} = 2.25 \text{ bita/simbolu},$$

što znači da je gori od koda fiksne dužine. U drugom slučaju imamo da je prosječna dužina kodne riječi za kod 3:

$$\bar{L} = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} = 1.75 \text{ bita/simbolu},$$

što znači da je u drugom slučaju kôd 3 bolji od koda fiksne dužine. Ovim smo efektivno pokazali da je kvalitet koda vezan za raspodjelu (vjerovatnoće) simbola ulaznog alfabeta.□

Definicija III.4. Pod **optimalnim** (ponekad se kaže **kompaktnim**) kodom podrazumijevamo jednoznačno dekodabilan kod sa najmanjom prosječnom dužinom kodne riječi za dati alfabet.□

Sada možemo da zaključimo da je cilj kodiranja izvora, u uslovima poznate raspodjele vjerovatnoća simbola ulaznog alfabeta, dizajniranje jednoznačno dekodabilnog koda koji je trenutan (radi jednostavnosti u postupku dekodiranja) i optimalan (radi minimizacije troškova – memorijskih i komunikacionih zahtjeva u procesu kodiranja). Prije nego pređemo na postupke dizajniranja traženog koda, uvedimo sljedeću važnu teoremu, koja se ponekad naziva **I Šenonovom teoremom** ili teoremom o kodiranju izvora. Teoremu formulišemo donekle nerigorozno, ali ipak sa dovoljno elemenata koji su potrebni za njeno razumijevanje.

Teorema III.3. Neka je dat alfabet sa n simbola sa poznatim vjerovatnoćama ovih simbola i entropijom $H(X)$. Niz od N simbola ovog alfabeta, gdje je $N \to \infty$, gdje su uzastopni simboli i.i.d., može se jednoznačno kodirati i dekodirati sa najmanje $NH(X)$ bita. U suprotnom smjeru kodiranje bez gubitaka nije moguće izvršiti sa manje od $NH(X)$ bita.□

I Šenonova teorema može se iskazati i na druge rigoroznije ili praktičnije načine. Najupečatljiviji praktični način da se iskaže ova teorema je sljedeća formulacija. Prosječna dužina kodne riječi jednoznačno dekodabilnog koda je veća ili jednaka entropiji simbola alfabeta koji se kodira:

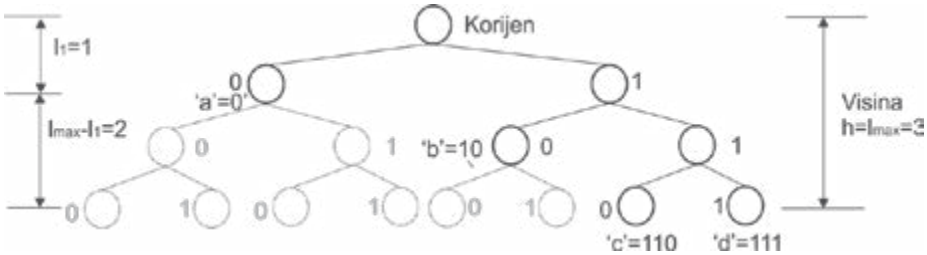
$$\bar{L} \geq H(X).$$

Prije nego dokažemo I Šenonovu teoremu, posmatrajmo sljedeću teoremu koja se naziva Kraftovom (ili rjeđe Kraft−MekMilanovom, engl. *McMillan*) nejednakošću.

Teorema III.4. Pretpostavimo da imamo ulazni alfabet sa m simbola. Prilikom kodiranja korišćen je D-arni izlazni alfabet (sa D simbola izlaznog alfabeta). Svaki jednoznačno dekodabilni trenutni kod zadovoljava sljedeću nejednakost:

$$\sum_{i=1}^{N} D^{-l_i} \leq 1,$$

gdje su $l_1, l_2, ..., l_N$ dužine kodnih riječi kojima su kodirani simboli ulaznog alfabeta.



Slika III.2. Ilustracija dokaza Kraftove nejednakosti. Čvor na visini $l_1 = 1$ eliminiše mogućnost pojave $D^{l_{\max}-l_1} = 2^{3-1} = 4$ listova

Dokaz. Već smo vidjeli da se trenutni kodovi mogu prikazati putem stabla, gdje su kodne riječi kojima se kodiraju simboli ulaznog alfabeta listovi. Putanja od korijena stabla do lista predstavlja dužinu kodne riječi l_j, $j \in [1,N]$. Neka je $l_{\max}$ dužina najduže kodne riječi, odnosno visina predmetnog stabla. Ukupan broj listova ovoga drveta ne može biti veći od $D^{l_{\max}}$. Posmatrajmo sada Sliku III.2. Ako imamo čvor na visini l_1 (rastojanje od korijena) od korijena, to eliminiše $D^{l_{\max}-l_j}$ čvorova koji bi se nalazili na nivou koji je na najvećoj udaljenosti $l_{\max}$. Ukupan broj čvorova koji je eliminisan na ovaj način je:

$$\sum_{j=1}^{N} D^{l_{\max}-l_j}$$

i ovaj broj je zasigurno manji od $D^{l_{\max}}$. Odavde slijedi:

$$\sum_{j=1}^{N} D^{l_{\max}-l_j} \le D^{l_{\max}}.$$

Član $D^{l_{\max}}$ može da se izdvoji ispred sume, nakon čega jednostavno slijedi dokaz teoreme, dijeljenjem obje strane sa $D^{l_{\max}}$:

$$D^{l_{\max}} \sum_{j=1}^{N} D^{-l_j} \le D^{l_{\max}} \Rightarrow \sum_{j=1}^{N} D^{-l_j} \le 1.\square$$

Kraftova nejednakost ima opšte važenje, odnosno važi i za jednoznačno dekodabilne kodove koji nisu trenutni. Dokaz (nešto složeniji) koji je dao MekMilan slijedi.

Dokaz*. Sada ćemo dokazati ovu teoremu za opšti netrenutni kod. Uvedimo sljedeću oznaku:

$$K^n = \left(\sum_{i=1}^{N} D^{-l_i} \right)^n = (D^{-l_1} + D^{-l_2} + ... + D^{-l_q})^n.$$

U sumi postoji N^n članova koji su oblika D^{-k}, gdje je:

$$k = l_{i_1} + l_{i_2} + ... + l_{i_n};$$

k je iz skupa prirodnih brojeva u intervalu od n do $nl_{\max}$. Označimo broj sabiraka u obliku D^{-k} sa N_k, pa možemo zapisati:

$$K^n = \sum_{k=n}^{nl_{\max}} N_k D^{-k}.$$

Za jednoznačno dekodabilan kod, N_k ne može da bude veće od broja različitih sekvenci dužine k od D različitih kodnih simbola, to jeste od D^k. Odavde slijedi:

$$K^n \leq \sum_{k=n}^{nl_{\max}} D^k D^{-k} = nl_{\max} - n + 1 < nl_{\max}.$$

Data relacija važi za svako n (npr. za veoma veliko n koje teži beskonačnosti). To znači da K mora da bude manje od 1:

$$K = \sum_{i=1}^{q} D^{-l_i} \leq 1,$$

čime je dokaz završen.□

Sada smo u prilici da, uz neka ograničenja, dokažemo iskaz I Šenonove teoreme. Radi lakšeg dokaza, pretpostavićemo sljedeće: (a) razmatraćemo samo trenutne kodove; (b) Kraftova nejednakost je svedena na jednakost i (c) dužine kodnih riječi mogu biti necjelobrojne!? Ovdje ne uvodimo nikakvo ograničenje na trenutne i netrenutne kodove pošto smo upravo vidjeli da zaključci moraju da važe jednako za obje klase kodova, dok su trenutni jedino pogodniji radi konstrukcije i dekodiranja. Predmetne pretpostavke ćemo relaksirati nakon što završimo sa dokazom.

Dokaz. Prosječna dužina kodne riječi je jednaka:

$$\bar{L}(l_1, l_2, ..., l_N) = \sum_{i=1}^{N} l(x_i) p(x_i) = \sum_{i=1}^{N} l_i p_i.$$

Dodajmo na ovu sumu Lagranžev množilac sa Kraftovom (ne)jedakošću:

$$\bar{L}(l_1, l_2, ..., l_N) = \sum_{i=1}^{N} l_i p_i + \lambda \left(\sum_{i=1}^{N} D^{-l_i} - 1 \right).$$

Parcijalna diferencijacija prosječne dužine po dužinama pojedinačnih kodnih riječi daje:

$$\frac{\partial \bar{L}(l_1, l_2, ..., l_N)}{\partial l_j} = p_j - \lambda D^{-l_j} \log D = 0 \Rightarrow p_j = \lambda D^{-l_j} \log D \qquad \forall j.$$

Sumirajući lijevu i desnu stranu prethodnog izraza po j, dobijamo:

$$\sum_{j=1}^{N} p_j = \lambda \log D \sum_{j=1}^{N} D^{-l_j} \Rightarrow 1 = \lambda \log D \Rightarrow \lambda = \frac{1}{\log D}.$$

Uočimo da smo u dokazu koristili ponovo Kraftovu nejednakost (sa jednakošću), te osobinu da je suma vjerovatnoća jednaka 1. Dakle, veza između vjerovatnoće simbola i dužine kodne riječi kojom se predmetni simbol kodira je:

$$p_j = \lambda D^{-l_j} \log D = \frac{1}{\log D} D^{-l_j} \log D = D^{-l_j} \Rightarrow l_j = -\log_D p_j \qquad \forall j,$$

pa je minimalna prosječna dužina kodne riječi:

$$\bar{L}(l_1, l_2, ..., l_N) = -\sum_{i=1}^{N} p(x_i) \log_D p(x_i) = -\sum_{i=1}^{N} l_i p_i = -\sum_{i=1}^{N} p_i \log_D p_i = H_D(X).$$

Dakle, ovim smo dokazali da pod datim uslovima (kodne riječi mogu imati necjelobrojne dužine i Kraftova nejednakost je svedena na jednakost) trenutni jednoznačno dekodabilni kodovi imaju prosječnu dužinu kodne riječi koja je sa donje strane ograničena entropijom sistema koji želimo kodirati.

Određivanjem drugih parcijalnih izvoda prosječne dužine kodne riječi po dužinama riječi i utvrđivanjem da su oni zasigurno pozitivni, jednostavno se može pokazati da je u pitanju minimum prosječne dužine kodne riječi:

$$\frac{\partial^2 \bar{L}(l_1, l_2, ..., l_N)}{\partial l_j^2} = -\lambda D^{-l_j} \log D = \lambda D^{-l_2} \log^2 D > 0 \qquad \forall j.$$

Prodiskutujmo kakve posljedice imaju neobična ograničenja (pretpostavke), koja smo uveli u dokaz teoreme, na njenu validnost, odnosno na tačnost iskaza teoreme. Pođimo od posljednje (a može se reći i najnelogičnije) pretpostavke da dužine kodnih riječi mogu biti necjelobrojne. Naravno, optimalne dužine kodne riječi $l_j = -\log_D p_j$ samo su u rijetkim slučajevima cijeli brojevi. Npr.: ako je $D = 2$ i $p_j = 0.25$, dobijamo $l_j = 2$, dok se u opštem slučaju dobijaju cjelobrojne vrijednosti optimalnih dužina kodne riječi samo kada je $p_j = D^{-r}$, gdje je r prirodni broj. U slučaju da nemamo ovako rijetku situaciju, možemo da kodiramo simbol sa dužinom kodne riječi dobijene zaokruživanjem l_j na veći cijeli broj: $l'_j = \lceil l_j \rceil$. Ova vrijednost je uvijek $-\log_D p_j = l_j \le l'_j \le l_j + 1 = -\log_D p_j + 1$, pa prosječna dužina kodne riječi, implementiranjem ove realnosti, postaje:

$$\bar{L}' \le \sum_{i=1}^{N} p(x_i) l'_j = -\sum_{i=1}^{N} p_i \log_D p_i + \sum_{i=1}^{N} p_i = H_D(X) + 1.$$

$$\bar{L}' \ge H_D(X).$$

Dakle, ovim se ne narušava kodna teorema, već se pokazuje da je prosječna dužina kodne riječi sa donje strane ograničena entropijom, a ujedno smo dobili i informaciju da postoji kodni postupak koji će dati prosječnu dužinu kodne riječi

koja je sa gornje strane ograničena na $H_D(X)+1$. Drugi uslov koji smo, prilikom dokazivanja ove teoreme, uveli je svođenje Kraftove nejednakosti na jednakost. Međutim, uvođenjem bilo kojeg drugog ograničenja:

$$\sum_{i=1}^{N} D^{-l_i} = \alpha$$

za $\alpha < 1$ ne mijenja dokaz teoreme. Bez obzira na usvojeno ograničenje, neophodno je da važi Kraftova nejednakost da bi bila zadovoljena I Šenonova teorema.

III.3 Pomoćni kodovi

Teorija informacija i kodova je oblast u kojoj se veoma često dešava da model informacije koju treba kodirati ili kanala preko kojega treba da se prenese ta informacija nisu u skladu sa pretpostavkama. Stoga se, u praksi, kodiranje izvora obavlja kombinacijom više postupaka. Slična situacija je i kod kodiranja kanala.

Ovdje smo sublimirali neke od najpoznatijih pomoćnih kodova koji se koriste za različite namjene kod kodiranja izvora i kanala: Grejov, RLE i diferencijalni kod.

III.3.1 Grejov kod

Grejov kod je dobio ime po svom pronalazaču Frenku Greju (engl. *Frank Gray*), koji ga je patentirao 1953. godine, za primjenu u telekomunikacijama. Kada kodiramo, na primjer, cijele brojeve putem binarne predstave, imamo situaciju da se dva susjedna broja, na primjer 7 i 8, predstavljaju četvorobitno, kao 0111 i 1000, odnosno da se razlikuju na sva četiri bita. Ono što je motivisalo Greja da razvije ovakav kod bila je činjenica što je prilikom slanja ovakvih informacija, koje se relativno malo razlikuju, bilo potrebno „opterećivati“ prekidačka kola tako da se mnogo stanja u svakom trenutku mijenja. Stoga je predložio postupak kodiranja kojim se obezbjeđuje da se bilo koje dvije susjedne brojne vrijednosti koje se kodiraju razlikuju samo na jednom bitu. Posmatrajmo sada mogućnost trobitnog kodiranja brojeva {0, 1, 2, …, 7}:

Brojna vrijednost	0	1	2	3	4	5	6	7
Binarni kod	000	001	010	011	100	101	110	111
Grejov kod	000	001	011	010	110	111	101	100.

Vidimo da se, u standardnoj binarnoj predstavi, brojne vrijednosti 1 i 2 razlikuju na drugoj i trećoj poziciji (kasnije ćemo u petom poglavlju ovo nazivati Hemingovom distancom koja je jednaka 2). Kod Grejovog koda se svaka susjedna kombinacija međusobno razlikuje samo na jednoj poziciji, a, što je takođe ovdje od značaja, samo na jednoj poziciji razlikuju se i posljednja i prva kombinacija. Ovo je, na primjer, od značaja kod uređaja kao što su brojčanici, gdje imamo cifru nižeg

reda i cifru višeg reda. U jednom trenutku, kada dođemo do maksimuma cifre nižeg reda, dalje uvećavanje daje povećanje cifre višeg reda, dok se cifra nižeg reda vraća na 0 (na primjer sa 09 inkrementiranjem dobijamo 10). U mnogim sistemima aktuelne vrijednosti su uglovi, pa se kod njih, takođe, mali i veliki uglovi suštinski malo razlikuju (npr. 0^o od 359^o) − po prethodnoj logici pogodno je da se malo razlikuju i po binarnom zapisu (u smislu broja pozicija na kojima se razlikuju njihove kodne riječi). U predmetnoj situaciji vidimo da se 0 (kodirana sa 000) u binarnoj predstavi razlikuje ponovo na jednom mjestu od 7 (kodirano sa 100). Postoji više mogućnosti (posebno za veće dužine kodne riječi) kako se ovo kodiranje može ostvariti, ali se postavlja pitanje o jedinstvenom postupku. Da bismo ovaj postupak opisali, uvedimo sljedeću notaciju. Bitove standardnog binarnog zapisa dekadnih brojeva označimo sa:

$$\{a_1, a_2, \ldots, a_n\},$$

gdje je n broj bita u binarnoj predstavi, a_1 je prvi bit (bit najveće važnosti), dok je a_n posljednji bit (bit najmanje važnosti). Slično definišimo i bite Grejovog koda, kao:

$$\{b_1, b_2, \ldots, b_n\}.$$

Ono što je odmah uočljivo, iz načina na koji smo kodirali trobitne poruke, jeste da su prvi biti u binarnoj poruci jednaki onima u kodu (ovo ne mora da važi u opštem slučaju formiranja Grejovog koda, ali je najpraktičnije i najčešće korišćeno):

$$a_1 = b_1.$$

Nešto je složenije da odredimo vezu između ostalih bita Grejovog koda i binarne predstave dekadnog broja. Premda formulu možemo napisati odmah, uradimo to postepeno. Pokušajmo da izrazimo b_2 na osnovu a_1 i a_2. Ovo je zgodno jer želimo da znamo drugi bit Grejovog koda kada primimo drugi bit binarnog koda, odnosno da ne odlažemo kodiranje. Prikažimo tabelu koja povezuje b_2 u funkciji a_1 i a_2:

$b_2 = f(a_1, a_2)$	$a_1 = 0$	$a_1 = 1$
$a_2 = 0$	0	1
$a_2 = 1$	1	0

Očigledno je u pitanju logička funkcija koja predstavlja jednostavnu ekskluzivno ili operaciju (ex-ili): $b_2 = a_1 \oplus a_2$. Ova relacija važi i za sve ostale bite u Grejovom kodu, pa možemo napisati:

$$b_i = a_{i-1} \oplus a_i \text{ za } i > 1.$$

Podsjetimo se da smo se već upoznali s operacijom ex-ili, te da smo je u dijelu s grupama i zapisivali korišćenjem operatora sabiranja +, imajući na umu o kakvoj se algebri radi. To i sada možemo raditi uz isti oprez kao i u prethodnim sluča-

jevima. Naravno, uvijek imamo i upit vezan za dekodiranje koda. Za prve bite u kodnoj riječi važi ponovo:

$$b_1 = a_1$$

i tu ne postoji ni najmanji problem. Kada dekodiramo naredne bite možemo koristiti podatak da nam je prvi bit dekodiran (odnosno da su dekodirani početni biti). Započnimo sa:

$$b_2 = a_1 \oplus a_2,$$

dodajmo a_1 po ex-ili operaciji na obje strane ove jednakosti:

$$a_1 \oplus b_2 = a_1 \oplus a_1 \oplus a_2.$$

Kod ex-ili operacije važi $0 \oplus 0 = 0$, odnosno $1 \oplus 1 = 0$, odnosno $a_1 \oplus a_1 = 0$. Ovo je prirodno jer je sabiranje kod ex-ili operacije isto što i oduzimanje. Ujedno $0 \oplus 0 = 0$ i $0 \oplus 1 = 1$, odnosno 0 je neutralni element koji ne mijenja rezultat u operaciji s drugim članom skupa, tako da je $0 \oplus a_2 = a_2$. Dakle, zaključujemo da:

$$a_2 = a_1 \oplus b_2,$$

odnosno da svaki naredni bit možemo da dekodiramo na osnovu prethodno dekodiranog bita, kao:

$$a_i = a_{i-1} \oplus b_i \text{ za } i > 1.$$

Zaključimo još jednom da Grejov kod nema efekat na prosječnu dužinu kodne riječi i da ona u procesu kodiranja ostaje ista, ali da se ovaj postupak iz različitih razloga često koristi u raznim oblicima kodiranja (ne samo kod kodiranja izvora), pa je stoga i ovdje naveden kao jedan od značajnih pomoćnih postupaka u kodiranju.

III.3.2 RLE kod

RLE kod (engl. *Run Length Encoding*) je postupak za kodiranje dužine uzastopnog broja istih simbola (istih vrijednosti). Posmatrajmo sljedeći misaoni eksperiment. Digitalna slika je matrica. Elementi matrice su osvjetljaji. Ako slikamo, na primjer, učionicu, dio zida na slici biće predstavljen osvjetljajima koji u jednom redu predstavljaju bijelo. Taj red će, u matrici, biti predstavljen putem istih vrijednosti. Stoga je uveden RLE kod kojim se veliki broj uzastopnih istih vrijednosti mijenja sa parom (vrijednost, broj ponavljanja). Pogledajmo sljedeći primjer. Neka je poruka:

12 12 12 12 12 7 7 7 8 8 8 8 8 8 1 1 1 1 1 1 1 2 2 2 2 16 16 16 16 16 16 9 7 7 7 7 7

RLE kod je: (12,5), (7,3), (8,6), (1,7), (2,4), (16,5), (9,1), (7,5).

Dakle, svako uzastopno ponavljanje simbola zamijenili smo samim simbolom i brojem koji predstavlja broj ponavljanja. Ukupan broj simbola u polaznoj poruci

bio je 36, dok je ukupan broj vrijednosti u RLE kodu bio 16. Ovdje možemo uvesti veličinu koja se naziva **stepen** ili **odnos kompresije** (engl. *Compression ratio*), koja predstavlja količnik potrebne memorije za nekomprimovanu poruku sa memorijom potrebnom za komprimovanu:

$$\mathrm{CR} = \frac{\mathrm{MEMORIJA}_{\mathrm{nekomprimovane}}}{\mathrm{MEMORIJA}_{\mathrm{komprimovane}}}.$$

U posmatranom primjeru kompresioni odnos iznosi $\mathrm{CR} = 36/16 = 9/4 = 2.25 = 225\%$, pod uslovom da su svi simboli kodirani istim brojem bita. Česta je situacija da se simboli alfabeta kodiraju jednim brojem bita, dok se dužine mogu kodirati manjim brojem bita. Na primjer, ako je u poruci sa N simbola koji se kodiraju sa n_s bita pronađeno K parova, a dužina ponavljanja se kodira sa n_k bita, onda je odnos kompresije:

$$\mathrm{CR} = \frac{Nn_s}{K(n_s + n_k)}.$$

Ponekad se kaže da smo kompresiju izvršili na 40%, što znači da je memorija komprimovane poruke 40% od memorije potrebne za nekomprimovanu. Očigledno, u pitanju je inverzija kompresionog odnosa 1/CR. Alternativna mjera uspješnosti kompresije je sačuvani prostor, odnosno koliko prostora smo sačuvali (u procentima) zahvaljući kompresiji. **Sačuvani prostor** (engl. *space saving*) je jednak:

$$\mathrm{SS} = 1 - \frac{\mathrm{MEMORIJA}_{\mathrm{komprimovane}}}{\mathrm{MEMORIJA}_{\mathrm{nekomprimovane}}} = 1 - \frac{1}{\mathrm{CR}}.$$

Posmatrajmo slučaj niza sa vrijednostima:

1 3 8 8 2 9 9 9 1 3 4 12 12 1 6 8 24 24 1 13 1 13 13 7.

RLE kod ove sekvence je:

(1,1), (3,1), (8,2), (2,1), (9,3), (1,1), (3,1), (4,1), (12,2), (1,1), (6,1), (8,1), (24,2), (1,1), (13,1), (1,1), (13,2), (7,1).

Dužina nekodirane riječi je 24, dok je broj parova u kodiranoj 18, pa su, pod pretpostavkom da su i vrijednosti i dužine predstavljeni istim brojem bita, kodni odnos i sačuvani prostor jednaki:

$$\mathrm{CR} = \frac{24}{2 \times 18} = \frac{24}{36} = \frac{2}{3} \qquad \mathrm{SS} == 1 - \frac{3}{2} = -\frac{1}{2}.$$

Ovo se naziva **negativnom kompresijom**, odnosno situacijom kada ono što bi trebalo biti komprimovano zauzima više memorije nego polazna poruka. Dakle, RLE kod nije pogodan za primjenu u svim situacijama, već samo onda kada se u

poruci dešava veliki broj istih uzastopnih vrijednosti. Ovakva situacija je rijetka u porukama, ali se kod nekih kodnih standarda dešava da u nekom transformisanom prostoru (npr. kod JPEG – engl. *Joint Photographic Expert Group* algoritma za kodiranje slike u prostoru diskretne kosinusne transformacije na koji je primijenjena kvantizacija) dobije veliki broj uzastopnih istih vrijednosti. Stoga se, kao što smo već i naglasili, ovaj kod rijetko primjenjuje samostalno, već se primjenjuje kao jedan korak u određenim algoritmima za kodiranje izvora.

Postoje brojne varijante RLE koda. Jedna od najpoznatijih je READ (engl. *Relative Element Address Designate*) kôd koji je bio namijenjen za slanje dokumenata putem telefonskih linija (faksova). Stranica se, kod faksova, šalje kao niz linija koje se sastoje od crnih i bijelih tačaka (piksela – *pixel*, skraćenica od engl. *picture element* – element slike). Jedna linija se malo razlikuje od naredne. Stoga je ideja ovog kodiranja, koje je doživjelo nekoliko varijanti, da se kod uzastopnih linija ne vrši ponovno kodiranje putem RLE koda, već da se samo pamte razlike između pojedinih linija.

III.3.3 Diferencijalni kod

Dešava se, na primjer, kod prenosa informacija koje predstavljaju mjerenja neke fizičke veličine (temperature, vlažnosti, pozicije broda ili aviona itd.) da se u uzastopnim mjerenjima dobijaju različite vrijednosti, koje se malo razlikuju. Ovakva poruka (mjerenje) je, kao što smo vidjeli, nepogodna za RLE kodiranje, ali se može primijeniti alternativna tehnika koja se naziva diferencijalnim kodom. U praksi postoji više oblika diferencijalnog kodiranja, a ovdje pomenimo samo rudimentarni postupak da se nakon prve vrijednosti, umjesto originalne poruke, šalju razlike susjednih vrijednosti.

Npr. ako je polazna poruka:

$$a_1\, a_2\, a_3\, a_4\, a_5 \ldots a_i\, a_{i+1} \ldots,$$

diferencijalni kod se može formirati kao:

$$a_1\, a_2 - a_1\, a_3 - a_2\, a_4 - a_3\, a_5 - a_4 \ldots a_i - a_{i-1}\, a_{i+1} - a_i \ldots$$

Posmatrajmo sljedeću poruku koja može predstavljati mjerenja temperature kroz 24 časa (započevši od jutarnjih časova):

23 25 27 29 31 32 33 33 34 33 32 32 31 29 27 26 26 25 24 24 23 22 23 23.

Diferencijalni kod formiran po prethodnom pravilu ove poruke je:

23 2 2 2 2 1 1 0 1 –1 –1 0 –1 –2 –2 –1 0 –1 –1 0 –1 –1 1 0.

Dekodiranje diferencijalnog koda obavlja se tako što se prepiše prvi primljeni simbol da bi se nakon toga svaki naredni određivao kao zbir prethodnog dekodiranog simbola sa razlikom:

$$a_1 = d_1,\ a_2 = d_2 + d_1,\ a_3 = d_3 + d_2,\ a_4 = d_4 + d_3,\ \ldots,\ d_i = a_i + a_{i-1},\ \ldots$$

Postoje i drugi tipovi kodova zasnovani na razlikama. Često se kod određenih tipova informacija ne vrši diferenciranje uzastopnih elemenata, već se računanje razlika provodi na većem broju simbola odjednom. Na primjer, kod video-signala za potrebe video-nadzora za očekivati je da su uzastopne slike u signalu (frejmovi) malo različite jedne u odnosu na druge. Stoga se prva slika u seriji kodira na jedan način, dok se za sve ostale računaju razlike uzastopnih slika. Ako su samo male razlike među uzastopnim slikama, onda će vrijednosti u slikama, koje predstavljaju razlike, biti mnogo manje nego u originalnim frejmovima, pa se često takvi frejmovi mogu kodirati mnogo efikasnije (sa većim stepenom kompresije) nego originalni.

III.4 Šenon–Fanoovo kodiranje*

Klod Šenon je bio izuzetan teoretičar. Dao je nemjerljiv doprinos teoriji informacija i kodova, uspostavljajući neke od fundamentalnih zakona u ovoj oblasti. Međutim, nije bio toliko kreativan kada se ima na umu dizajniranje kodova koji su u stanju da dostignu odgovarajući kvalitet (da se primaknu limitima predviđenim teorijom i teoremama).

Šenon−Fano kod za kodiranje izvora bio je pokušaj da se izvrši kodiranje izvora koje dostiže granice predviđene kodnom teoremom. U pitanju je bio postupak koji je u mnogo čemu predstavljao ad hok nesistematizovano rješenje. Stoga je veoma brzo odbačen, te danas kod kodiranja izvora nema primjenu, mada se neka njegova proširenja koriste u određenim algoritmima i postupcima kod složenih sistema dekodiranja. Ovdje ga navodimo iz istorijskih razloga, te smo ga i označili zvjezdicom. Uputno je upoznavanje s ovakvim postupcima jer se često u praksi poseže za sličnim ad hok rješenjima, koja su često limitirana i kriju brojne skrivene probleme.

Osnovna ideja kako doći do optimalnog (kompaktnog) koda za poznate vjerovatnoće greške pojedinih simbola kod ovog postupka (pa i kod nekih drugih) jeste uzeti dužinu kodne riječi koja je proporcionalna sa $-\log p_i$, gdje je p_i vjerovatnoća simbola. Kako se vjerovatnoće rijetko mogu pisati u formi D^k, gdje je k negativni cijeli broj, a D osnova logaritma, odnosno broj simbola kojim se vrši kodiranje izvora, rijetko možemo da posegnemo za kodiranjem kodnim riječima koje imaju takvu dužinu.

Dva primjera za ovakav tip kodiranja za binarni alfabet su:

	Kod	**(a)**		**Kod**	**(b)**	
S	p_i	$-\log_2(p_i)$	X_i	p_i	$-\log_2(p_i)$	X_i
s_1	1/4	2	00	1/2	1	0
s_2	1/4	2	01	1/4	2	10
s_3	1/4	2	10	1/8	3	110
s_4	1/4	2	11	1/8	3	111.

Prosječna dužina kodne riječi u oba slučaja dostiže optimalnu vrijednost, odnosno entropiju izvora:

$$H_a(S) = \log_2 4 = 2 \qquad L_a = 4 \times 0.25 \times 2 = 2 \text{ bita/simbolu}$$
$$H_b(S) = 1.75 \qquad L_b = 1.75 \text{ bita/simbolu.}$$

Odavde se nameće ideja da se za svaku poznatu vjerovatnoću kodne riječi odrede logaritmi za osnovu 2 kod binarnog alfabeta. U slučaju opšteg D-arnog alfabeta računao bi se logaritam s osnovom D. Dobijene vrijednosti bismo zaokružili na prvu veću cjelobrojnu i simbol kodirali tom dužinom kodne riječi. Na primjer:

S	p_i	$-\log_2(p_i)$	$\lceil -\log_2(p_i)+1 \rceil$	(a)	(b)
s_1	2/3	0.58	1	0	0
s_2	2/9	2.17	3	100	10
s_3	1/9	3.17	4	1010	11.

Dakle, ulazni alfabet je imao tri simbola s vjerovatnoćama {2/3, 2/9, 1/9}, koje smo kodirali kodom (a) koji ima dužine kodnih riječi $\lceil -\log_2(p_i)+1 \rceil$. Međutim, dobijeni rezultat nije optimalan, odnosno kod nije kompaktan, dok je kod označen sa (b) kompaktan i daje najmanju moguću prosječnu dužinu kodne riječi:

$$L_a = 1 \times (2/3) + 2 \times (2/9 + 1/9) = 4/3 \text{ bita/simbolu.}$$

Ovakav prost način očigledno ne vodi kompaktnim kodovima, pa se moramo poslužiti nekim sistematičnijim pristupom.

Šenon−Fano postupak ide logikom da se simboli u alfabetu dijele u dvije grupe sa *približno* polovinom vjerovatnoće, te da se jedna polovina kodira bitom 0, a druga polovina bitom 1. Postupak se dalje rekurzivno nastavlja na podijeljene skupove, do dobijanja skupa od jednog člana. Pogledajmo sada ponovo primjer:

S	p_i	I podjela	II podjela	III podjela
s_1	0.5	0	0	0
s_2	0.25	1	10	10
s_3	0.125	1	11	110
s_4	0.125	1	11	111.

U prvoj podjeli skup simbola je podijeljen na dva dijela: u prvom je samo jedan simbol (element) $\{s_1\}$, koji kodiramo sa 0 (izbor simbola kojim se vrši kodiranje je proizvoljan, samo mora biti različit od drugog skupa), dok se drugi skup sastoji od tri simbola $\{s_2, s_3, s_4\}$, koji su kodirani bitom 1. Drugi skup dijelimo u dva podskupa sa jednakim vjerovatnoćama: $\{s_2\}$, kodiran sa 0 (računajući prethodni kod to je 10), i $\{s_3, s_4\}$, kodiran sa 1 (računajući prethodni kod to je 11). Ova dva podskupa imala su iste vjerovatnoće. Konačno, simbol s_3 kodiramo sa 0 (odnosno, ukupno 110), dok simbol s_4 kodiramo sa 1 (ukupno 111). Predmetni postupak često daje kompaktan kod kao u prethodnom slučaju. Međutim, to ne mora da bude uvijek tako, jer kao što smo rekli začkoljica je u riječi *približno*, koju nije lako kvantifikovati.

Posmatrajmo sljedeći primjer:

S	p_i	I podjela	II podjela	III podjela	IV podjela
s_1	0.4	0	00	00	00
s_2	0.2	1	10	100	100
s_3	0.12	0	01	01	01
s_4	0.08	1	11	110	1100
s_5	0.08	1	11	110	1101
s_6	0.08	1	11	111	111
s_7	0.04	1	10	101	101.

Lako je uočiti da nije moguća podjela na podskupove sa tačno polovinom vjerovatnoće. Tako da u prvom krugu dijelimo simbole na dva skupa: $\{s_1, s_3\}$ s ukupnom vjerovatnoćom 0.52 (kodirani sa 0) i ostalih pet simbola, s ukupnom vjerovatnoćom 0.48 (kodirani sa 1). U drugoj podjeli kodiramo elemente prvog skupa sa 0 i 1 i tu prekidamo dalje tretiranje ovog skupa. Sada drugi skup možemo podijeliti u dva sa tačno po pola vjerovatnoća (0.24), i to $\{s_2,s_7\}$ (kodiran sa 1) i $\{s_4,s_5,s_6\}$ (kodiran sa 0). U trećem krugu kodiramo simbole skupa $\{s_2,s_7\}$, redom, sa 0 i 1, dok simbole drugog skupa moramo još jednom podijeliti da bismo prilikom četvrte podjele završili s kodiranjem. Prosječna dužina kodne riječi dobijena ovim postupkom je:

$$\bar{L} = 2 \cdot 0.52 + 3 \cdot 0.32 + 4 \cdot 0.16 = 3.64 \text{ bita / simbolu}.$$

Međutim, kod nije kompaktan, što je zapravo očigledno. Ako zamijenimo kod za simbole s_2 i s_3 i kodove za simbole s_4 i s_7, dobili bismo manju prosječnu dužinu kodne riječi, a kod bi zadržao osobinu jednoznačne dekodabilnosti.

Dakle, način podjele bitno utiče na kvalitet Šenon−Fano postupka. Ono što je dobra podjela u jednom koraku (približno iste vjerovatnoće u skupovima) može da bude problem u narednim koracima koji sprečava da se postigne kompaktni kod. Stoga se ovaj postupak često provodi tako što se pravi nekoliko podjela u

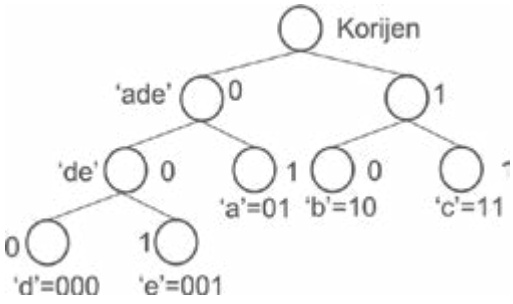
kritičnim djelovima algoritma i na kraju bira onaj skup kodnih riječi koje daju najmanju prosječnu dužinu. Čak ni takav postupak ne garantuje kompaktan kod, pa je vremenom Šenon−Fano algoritam postepeno zaboravljen i eventualno se koristi kao dio kodnih šema u nekim drugim kodnim postupcima (češće vezano za kodiranje kanala nego za kodiranje izvora).

III.5 Hafmenovo kodiranje

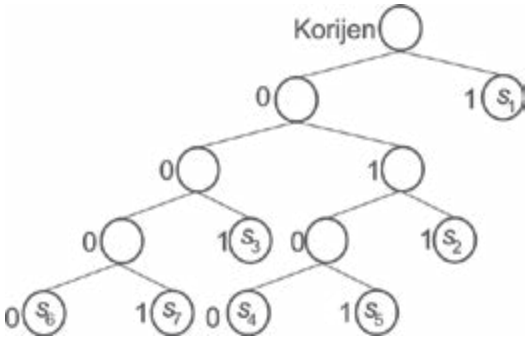
Kod formulisanja prve Šenonove teoreme srećna okolnost je ta što je kod koji je u stanju da postigne uslove ove teoreme relativno brzo bio razvijen, čime je dokazana mogućnost konstruisanja predmetnih kodova. Riječ je o Hafmenovim kodovima (dobili ime po Dejvidu Hafmenu, engl. *Davidu A. Huffman*u, koji ih je razvio kao student postdiplomskih studija na MIT-u, 1952. godine). Posmatrajmo sljedeći primjer.

Primjer III.4. Dat je alfabet sa pet simbola $X=\{a, b, c, d, e\}$, sa poznatim vjerovatnoćama pojavljivanja simbola $\{0.25, 0.25, 0.20, 0.15, 0.15\}$, respektivno. Kodirati simbole posmatranog alfabeta Hafmenovim kodom.

Rješenje: U svakom koraku kombinujemo dva najmanje vjerovatna simbola alfabeta, sabiramo njihove vjerovatnoće i jednom simbolu dodjeljujemo bit 0, a drugom simbolu bit 1. Na taj način formiramo novi generički simbol koji ima vjerovatnoću pojavljivanja koja je jednaka zbiru vjerovatnoća dva najmanje vjerovatna simbola. U prvom koraku (Slika III.3) grupišemo simbole *d* i *e*, prvom dodjeljujemo bit 0, a drugom bit 1 (dodjeljivanje bitova je proizvoljno i ne utiče na prosječnu dužinu kodne riječi koda niti na ostale performanse). Novi, generički, simbol ima združenu vjerovatnoću 0.3. Od preostala četiri simbola, biramo ponovo dva najmanje vjerovatna – simbole *b* i *c*, ponovo ih kodirajući sa 0 i 1, čime se dobija generički simbol s ukupnom vjerovatnoćom 0.45. U ovom koraku imamo tri simbola *a* i generičke kombinacije *bc* i *de* $\{a, bc, de\}$. Hafmenov postupak liči na dječju igru „muzičkih stolica“ (neizostavan dio svakog dječjeg rođendana), u kojoj u svakom krugu igračima ostaje na raspolaganju po jedna stolica manje. U narednom koraku



Slika III.3. Ilustracija drveta za Hafmenovo kodiranje



Slika III.4. Hafmenovo kodiranje alfabeta sa sedam simbola

objedinjujemo dva simbola *a* i generički *de* kao najmanje vjerovatna i konačno sabiramo u posljednjem koraku kombinacije *ade* i *bc*, dodjeljujući pojedinim granama kodove (bitove) 0 i 1.

Predmetnim postupkom dobijen je kod:

simboli:	*a*	*b*	*c*	*d*	*e*
kod:	01	10	11	000	001

sa strukturom stabla koja je prikazana na Slici III.3. Prosječna dužina kodne riječi koju smo dobili je:

$$\bar{L} = p_a l_a + p_b l_b + p_c l_c + p_d l_d + p_e l_e =$$
$$= 2 \cdot 0.25 + 2 \cdot 0.25 + 3 \cdot 0.20 + 3 \cdot 0.15 + 3 \cdot \mathbf{0}.15 = 2.3 \text{ bita/simbolu.}$$

Primjer III.5. Izvršiti Hafmenovo kodiranje alfabeta $S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ sa vjerovatnoćama {0.4, 0.2, 0.12, 0.08, 0.08, 0.08, 0.04}, respektivno. Ovaj alfabet je u prethodnoj sekciji kodiran Šenon−Fanoovim kodom.

Rješenje: Očigledna razlika između Hafmenovog i Šenon−Fano postupka je u redosljedu poteza. Kod Hafmena počinjemo od najmanje vjerovatnih simbola, objedinjujući prvo s_6 i s_7 u generički simbol s vjerovatnoćom 0.12. U narednom koraku objedinjujemo s_4 i s_5 u generički simbol s vjerovatnoćom 0.16. Dva najmanje vjerovatna simbola u narednom koraku su s_3 i generički s_6s_7, koji imaju združenu vjerovatnoću 0.24. Naredno „obuhvatanje" je s_2 i generički s_4s_5, s ukupnom vjerovatnoćom 0.36. U pretposljednjem koraku objedinjujemo generičke $s_3s_6s_7$ i $s_2s_4s_5$ i konačno imamo posljednje spajanje sa simbolom s_1.

Simboli ulaznog alfabeta su kodirani kao:

s_1 1 s_2 011 s_3 001 s_4 0100 s_5 0101
s_6 0000 s_7 0001.

Prosječna dužina kodne riječi koda jednaka je:

$$\bar{L} = 1 \cdot 0.4 + 3 \cdot (0.2 + 0.12) + 4 \cdot (0.08 + 0.08 + 0.08 + 0.04) =$$
$$= 0.4 + 0.96 + 1.18 = 2.48 \text{ bita / simbolu}$$

što je drastično bolje nego u slučaju Šenon−Fano postupka koji smo proveli. Entropija ovog alfabeta je jednaka:

$$H(S) \approx 2.42 \text{ bita/simbolu},$$

što pokazuje da je Hafmenov postupak produkovao prosječnu dužinu koja za nepuna 3% prevazilazi entropiju sistema.

Pretpostavimo da je poslata poruka $s_1 s_3 s_4 s_1 s_5 s_7 s_1$. Hafmenov kod ove poruke je 100101001010100011. Važno je uočiti da je Hafmenov kod trenutni – prefiksni (prikazan je kodnim stablom), odnosno da kada primimo posljednji bit kojim je neki simbol kodiran, možemo trenutno da ga dekodiramo. Dakle, $1 \rightarrow s_1$, $001 \rightarrow s_3$, $0100 \rightarrow s_4$, $1 \rightarrow s_1$, $0101 \rightarrow s_5$, $0001 \rightarrow s_7$, i $1 \rightarrow s_1$.□

Razumno pitanje koje se može postaviti je: Kako možemo sa sigurnošću da garantujemo da ne postoji postupak koji pod datim uslovima (poznavanjem vjerovatnoće pojavljivanja pojedinih simbola) može da produkuje manju prosječnu dužinu kodne riječi? Ovu dilemu nam razrješava naredna lema.

Lema III.1. Za svaku raspodjelu vjerovatnoća simbola alfabeta postoji optimalni trenutni kod s najmanjom prosječnom dužinom, koji zadovoljava sljedeće osobine:

1. Ako je $p_j > p_k$, onda $l_j \leq l_k$.
2. Dvije najduže kodne riječi imaju istu dužinu.
3. Dvije najduže riječi odgovaraju najmanje vjerovatnim simbolima i razlikuju se samo u posljednjem bitu.

Dokaz: Pretpostavimo da prvi stav ne važi. Međutim, ako $p_j > p_k$ može da znači $l_j > l_k$, onda taj kod ne bi bio optimalan, jer se prostom zamjenom kodova riječi za simbole s vjerovatnoćama p_j i p_k dobija kod s kraćom prosječnom dužinom kodne riječi. Dakle, prvi stav, po kontradikciji, mora da važi.

Ako drugi stav ne bi važio, onda bi postojala jedna najduža kodna riječ. Kako ta kodna riječ predstavlja list drveta, možemo da se vratimo jedan čvor unazad. Taj čvor ima samo jednog potomka (najdužu kodnu riječ), jer u suprotnom ne bi važilo da je ta riječ najduža (kada bi njen roditelj imao potomka u drugoj grani). Odnosno, mogli bismo da skratimo kodnu riječ za jedan, brišući posljednju granu i uzimajući da je kod zapravo roditeljski čvor. Dakle, ponovo smo po kontradikciji dokazali da drugi stav važi.

Treći stav implicira da su dvije najduže kodne riječi potomci istog čvora. Proces dokazivanja se obavlja na isti način kao prethodno, jer bismo u suprotnom mogli da skratimo obje najduže kodne riječi za jedan i da smanjimo prosječnu dužinu kodne riječi. Stoga možemo zaključiti da je i treći stav ovog iskaza tačan.□

Vidimo da Hafmenov postupak zadovoljava sva tri stava leme, odnosno da započinje najmanje vjerovatnim simbolima, da ti najmanje vjerovatni simboli imaju istu dužinu kodne riječi, te da im se kodovi razlikuju samo na posljednjem bitu.

III.5.1 Nebinarni Hafmenov kod

Modifikacija Hafmenovog koda je moguća i za slučaj nebinarnog izlaznog alfabeta (pod izlaznim alfabetom podrazumijevamo kod kojim se vrši kodiranje). U realizaciji ovog kodiranja postoji određeni problem, koji je najbolje opisati putem primjera.

Primjer III.6. Posmatrajmo skup simbola $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$ sa vjerovatnoćama $\{0.22, 0.2, 0.18, 0.15, 0.10, 0.08, 0.05, 0.02\}$, respektivno. Pretpostavimo da ovaj kod treba da kodiramo kvaternarnim izlaznim alfabetom sa simbolima $\{0, 1, 2, 3\}$. Cilj je, naravno, isti − dobiti najkraću moguću prosječnu dužinu kodne riječi.

Rješenje: Pođimo, kao što smo uradili do sada, obuhvatajući četiri najmanje vjerovatna simbola u jedan novi generički simbol s ukupnom vjerovatnoćom 0.25 (Slika III.5(a)). Zatim obuhvatimo dva najmanje vjerovatna simbola koja su preostala (to su $s_1 - s_4$) i zatim iskodirajmo dva dobijena generička simbola. Prosječna dužina kodne riječi u ovom slučaju je očigledna:

$$\bar{L} = 2 \text{ simbola izlaznog alfabeta/simbolu ulaznog alfabeta}.$$

Entropija ovoga sistema je:

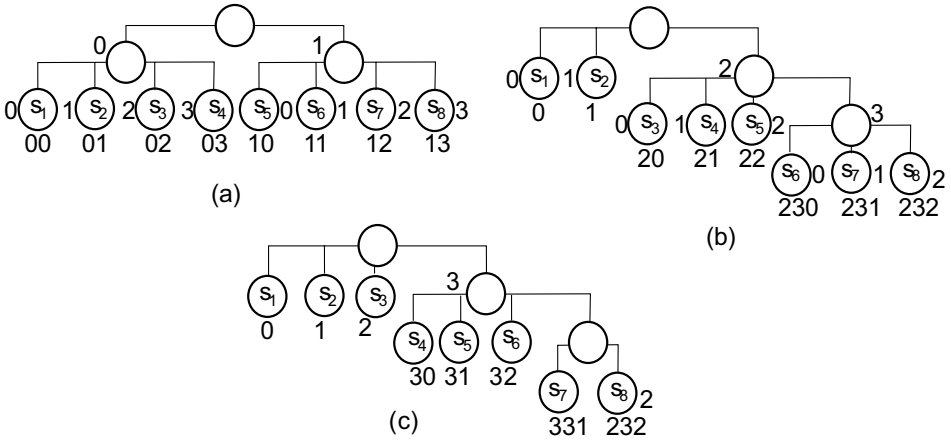
$$H(X) \approx 2.7535 \text{ bita}.$$

Dobili smo na prvi pogled nelogičan rezultat da je manja prosječna dužina kodne riječi od entropije, ali vrijedi istaći da poredimo na neki način babe i žabe, jer smo kod entropije radili s logaritmom s osnovom 2, dok je prosječna dužina kodne riječi računata za kvaternarni alfabet. Da bi ovo poređenje bilo fer, moramo onda računati entropiju s logaritmom s osnovom 4, pa dobijamo entropiju:

$$H_4(X) \approx 1.3767 \text{ simbola}.$$

Probajmo sada s obuhvatanjem tri simbola u prvom koraku (Slika III.5(b)) u generički simbol vjerovatnoće 0.15. U narednom koraku obuhvatimo četiri najmanje vjerovatna simbola i, konačno, u posljednjem koraku obuhvatimo tri simbola. Prosječna dužina kodne riječi u ovom slučaju je:

$$\bar{L} = 0.42 \cdot 1 + 0.43 \cdot 2 + 0.15 \cdot 3 = 1.73.$$



Slika III.5. Ilustrativni primjer nebinarnog Hafmenovog kodiranja

Dakle, ovim postupkom, u kojem obuhvatamo manji broj simbola u početnom koraku, dobijamo manju prosječnu dužinu kodne riječi. Nije teško zaključiti da je bitno da se na početku obuhvati što manji mogući broj simbola kako bismo u narednim koracima obuhvatili puni broj simbola koji odgovara broju simbola u izlaznom alfabetu. Stoga moramo da kreiramo postupak kojim bi odredili koliko simbola alfabeta treba da obuhvatimo u prvom koraku. Posmatrajmo ulazni alfabet (simbole koje treba kodirati) sa kardinalnošću N i izlazni alfabet kardinalnosti D. U posljednjem koraku treba da obuhvatimo D simbola i da od njih dobijemo jedan korijenski simbol; i da u svakom prethodnom koraku, osim prvog koraka, obuhvatimo D simbola i od njih kreiramo jedan novi generički simbol. Dakle, u svakom koraku „gubimo“ $D-1$ simbola, odnosno umanjujemo N sa po $D-1$, dok ne dobijemo broj manji ili jednak D, koji predstavlja broj simbola kodiranih u prvom koraku. Ovo se može opisati pseudokodom:

```
N'=N
WHILE N'>D
N'=N'-(D-1)
ENDWHILE
```

Sada možemo konačno da kodiramo naš alfabet koji ima $N=8$ simbola. Provodeći predmetni pseudokod, dobijamo da je $N'=2$, odnosno da u prvom koraku treba da obuhvatimo dva simbola. Provodeći ovaj postupak (Slika III.5(c)) dobijamo prosječnu dužinu kodne riječi:

$$\bar{L} = 0.60 \cdot 1 + 0.33 \cdot 2 + 0.07 \cdot 3 = 1.47 \text{ simbola}.$$

III.5.2 Neodređenost kod Hafmenovog kodiranja

U procesu Hafmenovog kodiranja može da se pojavi neodređenost, odnosno da imamo više kandidata za obuhvatanje simbola. Posmatrajmo navedeni problem kroz sljedeći primjer.

Primjer III.7. Dat je alfabet sa vjerovatnoćama simbola: {0.4, 0.2, 0.1, 0.1, 0.1, 0.05, 0.05}. Izvršimo Hafmenovo kodiranje binarnim alfabetom.

Rješenje: U prvom koraku obuhvatamo dvije riječi, tako da dobijamo skup sa vjerovatnoćama {0.4, 0.2, 0.1, 0.1, 0.1, **0.1**}**.** Posljednji simbol je nov, generički, nastao obuhvatanjem dva simbola. Postavlja se pitanje: koja dva od četiri simbola s istom vjerovatnoćom pojavljivanja da obuhvatimo u nastavku rada? Sa stanovišta prosječne dužine kodne riječi nema nikakve razlike. Međutim, ako sada obuhvatimo generički simbol sa nekim od tri preostala, sigurno ćemo dobiti da su najmanje vjerovatni simboli kodirani dužim riječima nego kada obuhvatimo dva nekodirana simbola. Kao što će biti pokazano u okviru urađenih zadataka, ipak je pogodnije u praksi obuhvatanje s onim riječima koje su do sada kodirane s manjim brojem bita. Postavlja se pitanje: zbog čega? U praksi se nakon kodiranja mogu dogoditi pogreške (više o greškama nakon kodiranja u narednim poglavljima), a te greške više utiču na situaciju kada su dužine kojima su kodirane pojedine riječi bitno nejednake. Za detalje pogledati riješene zadatke (zadatak 3.9).

III.5.3 Hafmenovi kodovi sa fiksnom dužinom kodne riječi

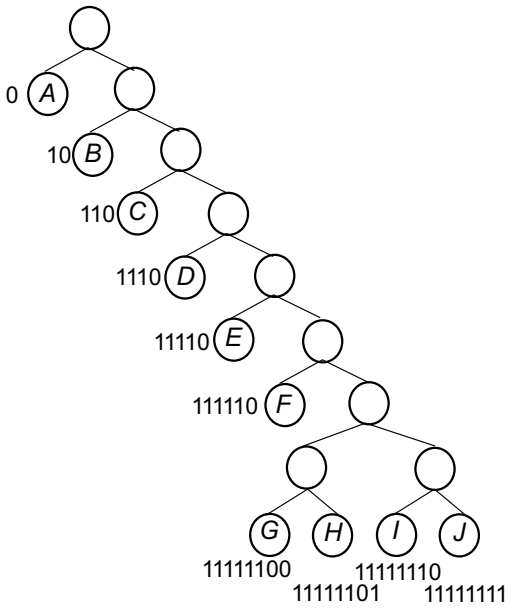
Često se u praksi Hafmenov kod ne primjenjuje za sve kodne riječi, već za određeni broj kodnih riječi koje se relativno često pojavljuju, dok se one kodne riječi koje se pojavljuju ekstremno rijetko kodiraju kodom fiksne dužine. Ovo stoga da se pojava nekih kodnih riječi, koje su inače ekstremno rijetke, ne bi odrazila na kvalitet i pokvarila kompresiju aktuelne poruke.

Primjer III.8. Posmatrajmo sljedeći primjer. Kod ima 10 simbola, od kojih se posljednja 4 javljaju sa jako malim vjerovatnoćama:

A	0.4	*B*	0.25	*C*	0.15	*D*	0.10	*E*	0.05
F	0.04	*G*	0.004	*H*	0.003	*I*	0.002	*J*	0.001.

Rješenje: Ovaj kod možemo kodirati standardnom Hafmenovom tehnikom (Slika III.6). Dobili smo prosječnu dužinu kodne riječi:

$$\bar{L} = 1 \cdot 0.40 + 2 \cdot 0.25 + 3 \cdot 0.15 + 4 \cdot 0.10 + 5 \cdot 0.05 + 6 \cdot 0.04 + 8 \cdot 0.01 =$$
$$= 2.32 \text{ bita/simbolu.}$$



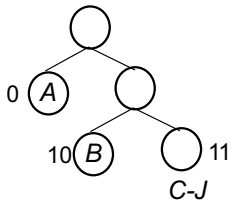
Slika III.6. Alfabet sa rijetkim simbolima kodiran standardnim Hafmenovim kodom

Simbole smo kodirali sljedećim kodnim riječima:

A	0	*B*	10	*C*	110	*D*	1110
E	11110	*F*	111110	*G*	11111100	*H*	11111101
I	11111110	*J*	11111111.				

U pitanju je kompaktan (optimalni) kod i nije moguće dobiti kod sa manjom prosječnom dužinom kodne riječi, ali imamo problem što nam se pojavljuju četiri simbola koja su kodirana s izuzetno dugačkim kodnim riječima. To dalje dovodi do problema da kada se u nekoj poruci pojavi više rijetkih simbola, može doći do negativne kompresije. Pored toga, u slučaju grešaka u prenosu poruke, nepogodne su predugačke kodne riječi jer se onda takav kod teško ispravlja. Jedan od načina da se obavi ovo kodiranje je da se grupišu najrjeđe kodne riječi u skup koji ima 2^r simbola, te da se taj generički simbol kodira na uobičajeni način fiksnom dužinom kodne riječi. Pogledajmo ilustraciju na Slici III.7, kada je $r=3$, odnosno kada najrjeđih osam simbola tretiramo kao jedan simbol (možete provjeriti da se s obuhvatanjem $r=2$, odnosno 2^r najmanje vjerovatnih simbola ništa posebno ne mijenja u ovoj situaciji u odnosu na standardno Hafmenovo kodiranje). U slučaju $r=3$, simboli *C*-*J* se pojavljuju s ukupnom vjerovatnoćom 0.35.

Dakle, simboli *A* i *B* su kodirani na standardni način sa 0, odnosno 10, respektivno, dok je generički simbol *C*-*J* kodiran sa 11. Ovo 11 se tretira kao prefiks pojedinač-



Slika III.7. Alfabet sa rijetkim simbolima kodiran Hafmenovim kodom sa fiksnom dužinom kodne riječi

nih kodnih riječi. Pošto je 2^r simbola u ovom generičkom, onda svaki kodiramo sa sufiksom od dodatnih r bita:

C 11$\underline{000}$ D 11$\underline{001}$ E 11$\underline{010}$ F 11$\underline{011}$
G 11$\underline{100}$ H 11$\underline{011}$ I 11$\underline{110}$ J 11$\underline{111}$,

gdje je sufiks fiksne dužine vizuelno naznačen podvučenim slovima. Prosječna dužina kodne riječi u ovom slučaju je:

$$\bar{L}' = 1 \cdot 0.40 + 2 \cdot 0.25 + 5 \cdot 0.35 = 2.65 \text{ bita}.$$

Dakle, činjenicu da želimo da ograničimo kod kojim kodiramo najmanje vjerovatne kodne riječi plaćamo u ovom slučaju povećanjem prosječne dužine kodne riječi od 0.33 bita.

III.5.4 Hafmenovi kodovi kod i.i.d. izvora

Ako imamo poruke sa i.i.d. simbolima, možemo da ostvarimo dobit kodiranjem više uzastopnih simbola. Ilustrujmo ovo kroz jednostavan primjer.

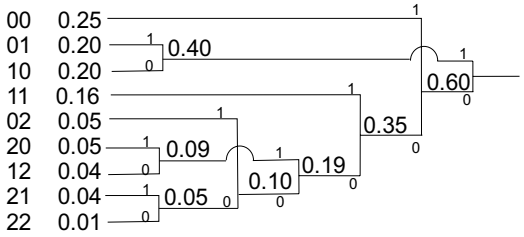
Primjer III.9. Posmatrajmo ternarni alfabet {0, 1, 2} sa vjerovatnoćama simbola {0.5, 0.4, 0.1}. Izvršiti Hafmenovo kodiranje i odrediti prosječnu dužinu kodne riječi. Zatim, pod pretpostavkom da su u pitanju i.i.d. simboli, izvršiti Hafmenovo kodiranje grupa od po 2, odnosno 3 uzastopna simbola. Izvršiti poređenje s entropijom alfabeta, odnosno sa I Šenonovom teoremom.

Rješenje: Prva situacija je krajnje jednostavna: najvjerovatniji simbol će biti kodiran jednim bitom, dok će druga dva biti kodirana sa dva bita. Prosječna dužina kodne riječi je:

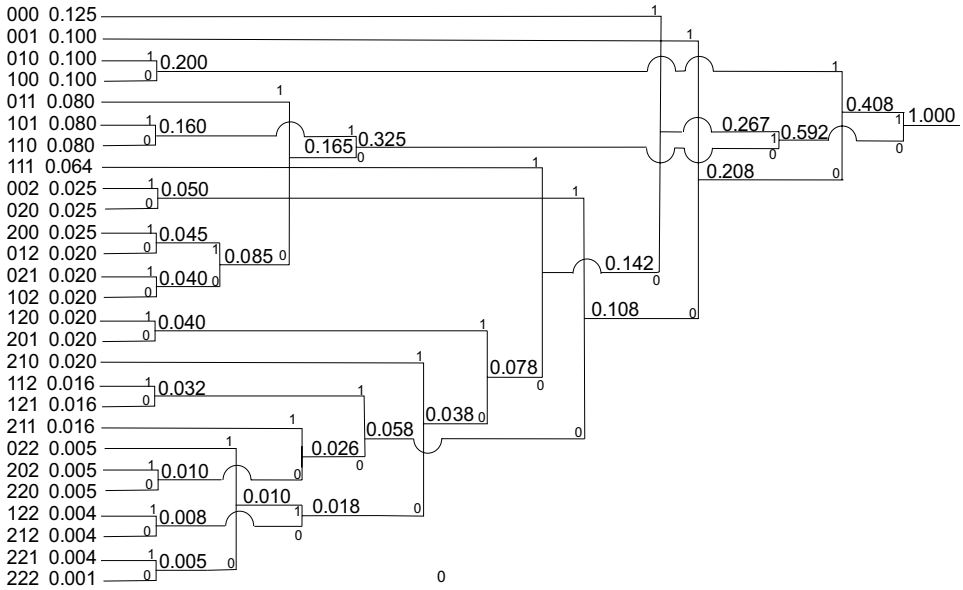
$$\bar{L} = 1 \cdot 0.5 + 2 \cdot 0.5 = 1.5 \text{ bita}.$$

Entropija je:

$$H(X) = -0.5 \log_2 0.5 - 0.4 \log_2 0.4 - 0.1 \log_2 0.1 \approx 1.361 \text{ bita/simbolu}.$$



Slika III.8. Hafmenovo stablo za kodiranje parova simbola kod i.i.d. sistema



Slika III.9. Hafmenovo stablo za kodiranje trojki simbola kod i.i.d. sistema

Rezultat očigledno nije loš, ali imamo gubitak od 0.139 bita po simbolu u odnosu na entropiju, mada predmetnim postupkom i ne možemo postići bolji rezultat. Formirajmo sada alfabet koji se sastoji od grupa od po dva simbola:

$$\{00, 01, 02, 10, 11, 12, 20, 21, 22\},$$

s vjerovatnoćama:

$$\{0.25, 0.20, 0.05, 0.20, 0.16, 0.04, 0.05, 0.04, 0.01\}.$$

Na Slici III.8 prikazano je kodno stablo za ovaj slučaj. Kodne riječi kojima se kodiraju parovi simbola su: $c(00) = 01$, $c(01) = 11$, $c(10) = 10$, $c(11) = 001$, $c(02) = 00001$, $c(20) = 00011$, $c(12) = 00010$, $c(21) = 000001$, $c(22) = 000000$.

Prosječna dužina kodne riječi u ovom slučaju (ali sada upotrijebljena za kodiranje para simbola) je:

$$\bar{L}_2 = 2 \cdot 0.65 + 3 \cdot 0.16 + 5 \cdot 0.14 + 6 \cdot 0.05 = 2.78 \text{ bita} / 2 \text{simbola}.$$

Kodna teorema je i dalje zadovoljena pošto je entropija za kodiranje dva međusobno nezavisna simbola:

$$2H(X) \approx 2.722 \text{ bita}.$$

Dakle, za kodiranje jednog simbola sada je potrebno:

$$\bar{L} = \bar{L}_2 / 2 = 1.39 \text{ bita/simbolu}.$$

Ovim smo gubitak, u odnosu na entropiju pojedinačnog događaja, sveli na 0.029 bita po jednom simbolu, što je veoma značajno kada poredimo s kodiranjem pojedinačnih simbola.

Ponovimo postupak za kodiranje tri simbola. U skupu onda imamo po jednu vjerovatnoću {0.125, 0.064, 0.001}, zatim imamo po tri vjerovatnoće {0.1, 0.08, 0.025, 0.016, 0.005, 0.004} i šest vjerovatnoća 0.02. Na Slici III.9 prikazano je navedeno kodno stablo. Prosječna dužina kodne riječi za kodiranje tri simbola je:

$$\bar{L}_3 = 3 \cdot 0.425 + 4 \cdot 0.304 + 5 \cdot 0.05 + 6 \cdot 0.193 + 7 \cdot 0.005 + 8 \cdot 0.023 \approx 4.118 \text{ bita} / 3 \text{ simbola}$$

$$\bar{L} = \bar{L}_3 / 3 \approx 1.3727 \text{ bita/simbolu}.$$

Dakle, nastavlja se trend popravke koda, odnosno smanjivanje prosječne dužine kodne riječi, ali daleko od spektakularnog (što se, zapravo, može i očekivati kada imamo na umu da smo već prilikom kodiranja sa dva simbola ostvarili rezultat koji je veoma blizak entropiji). Očigledno ovaj progres plaćamo povećanjem zahtjeva u procesu kodiranja.

III.5.5 Hafmenovi kodovi kod Markovljevih sistema

Kod Markovljevog sistema prvog reda važi:

$$H(X,Y) = H(X) + H(Y \mid X) \le H(X) + H(Y),$$

$$\frac{H(X,Y)}{2} \le \frac{H(X) + H(Y)}{2} \le H(X),$$

ako je $H(Y) \text{ " } H(X)$. Prilikom kodiranja više uzastopnih simbola Markovljevog sistema, zadovoljena je I Šenonova teorema $\bar{L}_2 \ge H(X,Y)$. Međutim, prosječna dužina kodne riječi, normalizovana po broju simbola, $\bar{L}_2 / 2$, može da bude manja od entropije polaznog događaja $H(X)$. Na osnovu ovoga zaključujemo da je kod Markovljevih sistema moguće ostvariti veću uštedu, u smislu prosječne dužine

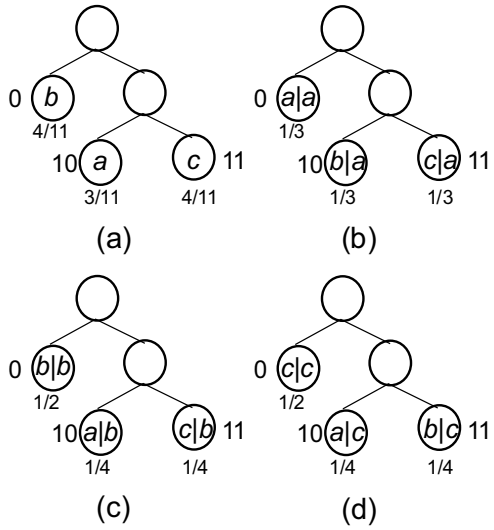
kodne riječi, pravilnom primjenom Hafmenovog postupka na veći broj simbola, nego što je to u slučaju i.i.d. sistema. Metodologiju je najlakše ilustrovati kroz primjer.

Primjer III.10. Posmatrajmo Markovljev sistem dat na Slici II.4. Odrediti prosječnu dužinu kodne riječi kada se primijeni Hafmenov postupak na pojedinačne simbole, te ako se primijeni, imajući na umu da je u pitanju Markovljev sistem prvog reda. Kodirati i dekodirati putem oba postupka riječ: *bababacaabc*.

Rješenje: Vjerovatnoće simbola u stacionarnom režimu su $p(a) = 3/11$, $p(b) = 4/11$, $p(c) = 4/11$ (pogledati Primjer II.2). Primjenom Hafmenovog koda na ovaj alfabet (Slika III.10(a)) dobijamo kod $\mathbf{c}(a) = 10$, $\mathbf{c}(b) = 11$, $\mathbf{c}(c) = 0$, gdje smo sa $\mathbf{c}()$ označili odgovarajuće kodno preslikavanje. Prosječna dužina kodne riječi je:

$$\bar{L} = 2 \cdot \frac{7}{11} + 1 \cdot \frac{4}{11} = \frac{18}{11} \text{ bita/simbolu}.$$

Međutim, imajući na umu da se radi o Markovljevom sistemu, možemo da kodiramo riječi tako da primijenimo saznanje da je prethodni simbol bio određene vrijednost i da kodiramo naredni simbol uz predznanje o prethodnom simbolu. Na Slici III.10 (b)−(d) date su šeme Hafmenovog kodiranja kada znamo da su respektivno prethodni simboli bili *a*, *b* i *c*. Prosječne dužine kodnih riječi za ove tri tabele su:



Slika III.10. Hafmenovo kodiranje Markovljevog ternarnog sistema I vrste: (a) Za vjerovatnoće u stacionarnom stanju; (b) Za vjerovatnoće u zavisnosti (nakon) simbola 'a'; (c) Za vjerovatnoće u zavisnosti (nakon) simbola 'b'; (d) Za vjerovatnoće u zavisnosti (nakon) simbola 'c'

$$\bar{L}_a = 1\cdot\frac{1}{3}+2\cdot\frac{2}{3}=\frac{5}{3} \qquad \bar{L}_b=\bar{L}_c=1\cdot\frac{1}{2}+2\cdot\frac{1}{2}=\frac{3}{2}.$$

Prosječna dužina kodne riječi je sada:

$$\bar{L}=\frac{3}{11}\cdot\frac{5}{3}+\frac{4}{11}\cdot\frac{3}{2}+\frac{4}{11}\cdot\frac{3}{2}=\frac{17}{11}\text{ bita/simbolu}.$$

Vidimo da smo ostvarili uštedu od 1/11 bita po simbolu, odnosno nešto oko 5.56%, što nije beznačajno. Da smo provodili postupak kao da su u pitanju i.i.d. procesi, ostvarili bismo (provjerite) prosječnu dužinu kodne riječi od ≈ 1.5868 bita/simbolu, što je za oko 0.041 bit po simbolu više (uz istu složenost) nego kada koristimo infomaciju da je u pitanju Markovljev sistem I reda. Posmatrajmo sada poruku:

abcbccabc.

Prvi simbol se kodira, na primjer, putem tabele za vjerovatnoće u stacionarnom režimu (jer nema prethodnika) (Slika III.10 (a)) kao 10, zatim se simbol *b* kodira na osnovu tabele sa saznanjem da je prethodni simbol *a* kao 10, zatim imamo *c* nakon *b* (Slika III.10(c)) 11, nakon toga *b* nakon *c*, što se također kodira kao 11, ponovo *c* u zavisnosti od *b* 11, zatim *c* nakon *c* kodira se kao 0 itd. Konačna poslata sekvenca je:

10 10 11 11 11 0 10 10 11.

Dekodiranje se obavlja na isti način kao i kodiranje. Započinjemo od tabele sa simbolima kojima ništa ne prethodi (Slika III.10(a)), primamo 10 i to dekodiramo kao *a*, a zatim koristimo šemu za dekodiranje u zavisnosti od *a* (Slika III.10(b)) kada primljeno 10 dekodiramo kao *b* itd. Dovršite sami proces kodiranja i dekodiranja u ovom slučaju.□

Predmetni postupak nameće obavezu da koder i dekoder imaju kod formiran na isti način. Jedan način je da koder pošalje **rječnik**, odnosno spisak simbola i kodnih riječi dekoderu, što ponekad može da optereti komunikaciju, posebno kod kratkih poruka kada rječnik može biti veći nego je ostvarena ušteda u procesu kodiranja. Očigledno se radi o mani Hafmenovog postupka, o čemu će biti više riječi kasnije.

III.5.6 Adaptivni i dinamički Hafmenovi kodovi

Očigledan problem Hafmenovog koda je rječnik, kao i potreba da se znaju vjerovatnoće simbola (ili uslovne vjerovatnoće). Posebno problem narasta kada se desi promjena vjerovatnoće simbola tokom komuniciranja, pa je potrebno nekoliko puta slati kodne knjige. Ovo usporava komunikaciju i smanjuje uštedu koju postižemo procesom kodiranja, odnosno kompresije. U slučaju kada komunikacija podrazumijeva podatke čija je statistika unaprijed dobro poznata, ovakav problem

ne postoji. Kodne knjige – kodovi – rječnici formiraju se samo jednom ili rijetko. Međutim, postoji mnogo situacija kada to nije slučaj. Ovo je razlog zbog kojeg je u praksi razvijen veliki broj adaptivnih ili dinamičkih Hafmenovih kodova sa različitim karakteristikama. Jedna popularna strategija je da se počne s inicijalnim vjerovatnoćama pojavljivanja svakog simbola:

$$\{p_1^{(0)}, p_2^{(0)}, ..., p_N^{(0)}\},$$

i na strani prijema i na strani predaje. Kodovi se na obje strane formiraju tako što se na isti način vrši kodiranje (na isti način se određuje koji će se bit pridružiti kojoj grani u kodnom stablu). Ako ne postoji predznanje o vjerovatnoćama, može se uzeti da su simboli na početku jednako vjerovatni. Dalje se propušta relativno veliki broj simbola koji se kodiraju početnim Hafmenovim kodom. Neka je taj broj simbola u bloku K. Za svaki simbol se izbroji koliko puta se pojavio:

$$\{k_1^{(1)}, k_2^{(1)}, ..., k_N^{(1)}\}.$$

Na osnovu ovog broja simbola u bloku, vrši se ažuriranje vjerovatnoća:

$$p_i^{(1)} = \alpha p_i^{(0)} + (1-\alpha)\frac{k_i^{(1)}}{K}.$$

Parametar α se bira u granicama [0,1], gdje manje vrijednosti daju prednost izbrojanom u odnosu na pretpostavljeno. Hafmenovo kodiranje se sada vrši na osnovu ažuriranih vjerovatnoća:

$$\{p_1^{(1)}, p_2^{(1)}, ..., p_N^{(1)}\},$$

kao i dekodiranje na strani prijema, uz ista, unaprijed dogovorena pravila. Postupak se nastavlja za naredne blokove podataka.

Predmetni postupak ima određene nedostatke, od kojih je jedan potreba za preračunavanjem kodnog stabla, a drugi usporavanje procesa u trenutku kada se vrši novo računanje kodne knjige. Međutim, oba ova nedostatka u današnjim sistemima nisu toliko kritična kao treći nedostatak, a to je velika osjetljivost na bilo kakvu grešku u procesu. Mala pogreška u procesu određivanja ažuriranih vjerovatnoća može da donese katastrofalne greške u procesu dekodiranja. Te pogreške mogu da budu i u prenosu, pa čak i usljed ograničene dužine registara kod aritmetičkih operacija provedenih na različitim procesorima na prijemu i predaji (različitog zaokruživanja rezultata).

Problemi kod gore opisanog intuitivnog algoritma prouzrokovali su dalji razvoj ove oblasti. Danas se pod najboljim dinamičkim algoritmom za Hafmenovo kodiranje podrazumijeva onaj koji je predložio Viter (engl. *Vitter*), negdje krajem 1970-ih godina. Drugi poznati algoritam za ovu namjenu je FGK – Foler–Galager–Knutov

(engl. *Faller–Gallager–Knuth*). Napominjemo da je Galager dao mnoge bitne doprinose u teoriji informacija i kodova, dok je Knut jedan od najznačajnijih naučnika u oblasti kompletne teorije algoritama.

Oba pomenuta algoritma su po svojoj strukturi složenija nego što dozvoljava opis ovoga poglavlja. Ujedno, jedan od ključnih problema Hafmenovog koda, vezan za složenost kod Markovljevih sistema višeg reda, ovim nije razriješen.

III.6 Lempel–Ziv–Velč kodovi

Imajući na umu osnovne probleme Hafmenovog kodiranja: problem određivanja uslovnih vjerovatnoća za Markovljeve sisteme višeg reda, suboptimalnost procesa kodiranja kada se Markovljev sistem višeg reda kodira pod pretpostavkom da je u pitanju sistem nižeg reda (suboptimalnost podrazumijeva veću prosječnu dužinu kodne riječi u odnosu na onu koja se može postići), složenost kada se radi sa sistemima višeg reda i konačno problem prenošenja rječnika, postojala je jasna potreba da se razviju kodovi koji ove probleme prevazilaze.

U ovoj sekciji obradićemo nekoliko varijanti Lempel–Ziv kodova (dobili ime po Avramu Lepmelu i Jakovu Zivu, engl. *Abraham Lempel* i *Jacob Ziv*), odnosno Lempel–Ziv–Velč (LZW) (po Teriju Velču, engl. *Terry Welch*) koda. Ovi kodovi pripadaju široj familiji kodova zasnovanih na rječniku (engl. *dictionary based codes*). Kod ovih kodova, rječnik se ne prenosi, već se formira na odgovarajući način i na strani predaje i na strani prijema.

Hafmenov kod (postupak) je optimalan pod uslovom poznatih vjerovatnoća. LZ(W) postupak optimalan je u uslovima kada su vjerovatnoće nepoznate (odnosno kada se i ne žele odrediti). Teorijski okvir će biti opisan u narednoj podsekciji, zatim dajemo osnove LZ77 i LZ78 kodiranja, a na kraju dajemo ono unapređenje koje je predložio Velč i koje LZW postupak čini najboljim.

III.6.1 LZ postupak – teoreme*

Algoritamski postupak LZ kodiranja slijedi iz nekoliko teorema i lema koje su prilično složene i za razumijevanje kao i za dokazivanje. Stoga su ovdje priložene bez dokaza. Zainteresovani čitaoci dokaze mogu pronaći u literaturi, s tim što treba znati da su u ove četiri dekade, koje su slijedile nakon originalnih radova Lempela i Ziva, ipak izvršena određena pojednostavljenja postupka dokazivanja, te da se mogu naći i obuhvatniji i jednostavniji dokazi.

Lempel i Ziv definišu pojam **fraze**. Fraza je određeni dio kodne riječi koji predstavlja nešto što se ponavalja u originalnoj poruci. Broj fraza se označava kao C_n i on korespondira sa brojem riječi u našem rječniku. Očigledno, ako znamo da

je broj fraza C_n, možemo ih kodirati sa $\log C_n$ simbola. Radi pojednostavljenja, smatrajmo da je alfabet u kojem se vrši kodiranje binaran. Kada kodiramo simbole, uzimamo frazu koju kodiramo sa $\log C_n$ bita, koju nazivamo **prefiks**, na koji dodajemo sufiksni bit po kojem se ovaj dio koda razlikuje od bilo koje riječi koja je do tog momenta upisana u rječnik. Dakle, za kodiranje dijela sekvence, koja se do trenutka kada imamo C_n fraza nije pojavljivala u rječniku, treba nam $\log C_n + 1$ bit. Naravno, sa kodiranjem bilo koje nove fraze inkrementiramo i broj C_n.

Teorema III.5. Pretpostavimo da imamo ergodični proces. Neka je C_n broj fraza u poruci navedenog procesa koja je izdijeljenja u n nepreklapajućih blokova. Sljedeća relacija:

$$\max\left[\lim_{n\to\infty}\frac{C_n[\log C_n+1]}{n}\right]\le h$$

zadovoljena je sa vjerovatnoćom 1. Ovdje je sa h označena entropija sekvence od n simbola, podijeljenja sa brojem simbola (suštinski prosječna dužina kodne riječi koja se može postići Hafmenovim postupkom). □

Šta teorema suštinski znači? Ako formiramo kod sa frazama koje su kodirane putem prefiksa na navedeni način, pod uslovom da imamo ergodičan proces, ostvarićemo uštedu u kodiranju koja je limitirana entropijskim odnosom h. Da bismo ostvarili predmetnu uštedu, broj fraza u sistemu mora da bude ograničen, jer ćemo u suprotnom fraze kodirati velikim brojem bita, pa će nestati ušteda. O ovome govori sljedeća lema.

Lema III.2. Broj fraza u bilo kakvoj podjeli ergodične sekvence u nepreklapajuće podskupove zadovoljava sljedeću nejednakost:

$$\lim_{n\to\infty}\frac{C_n\log n}{n}\le 1.$$

Dakle, broj fraza je, grubo govoreći, asimptotski (kada n teži beskonačno) ograničen sa $n/\log n$, a to dalje znači da je potrebno n bita za kodiranje svake fraze (u najgorem slučaju). Naredna lema je posebno značajna i naziva se Zivovom (čime se i naglašava činjenica da je njegov doprinos razvoju koda veći nego je to doprinos Lempela).

Lema III.3. (Zivova lema). Pretpostavimo da je sekvenca $(x_1, x_2, ..., x_n)$ parsirana (podijeljena u nepreklapajuće podstringove) od C_n različitih fraza $(Y_1, Y_2, ..., Y_{C_n})$. Sa W_i označimo k bita koji prethode Y_i. Sa C_n^{lw} označimo broj fraza koje imaju dužinu l i $W_i = w$. Tada važi sljedeća nejednakost:

$$-\log P(X_1, X_2, ..., X_n \mid W_1) \ge \sum_l \sum_n C_n^{lw} \log C_n^{lw}.$$

Lema III.4. Neka je L nenegativna cjelobrojna slučajna promjenljiva s očekivanjem μ_L. Entropija ove slučajne promjenljive zadovoljava ograničenje:

$$H(L) \leq (\mu_L + 1)\log(\mu_L + 1) - \mu_L \log \mu_L.$$

Premda, pored prikazanih, postoje i druge teoreme, leme i stavovi koji se odnose na LZ kodiranje, ovim ćemo zaključiti naš pregled formulacija koje pokazuju i dokazuju optimalnost LZ postupka, kao i njegovu univerzalnost. Dakle, ovi stavovi govore o broju fraza, broju bita potrebnom za kodiranje fraza, povezuju broj fraza s prefiksom poruke, te, konačno, uspostavljaju granice entropije dužine kodne riječi. Neki od ovih stavova imaju univerzalniju primjenu koja izlazi van okvira LZ koda (kao, na primjer, Lema III.4, kojom se pokazuje opšte pravilo kod nenegativnih cjelobrojnih slučajnih promjenljivih).

III.6.2 LZ postupak

Sada ćemo se upoznati s osnovnim postupcima koji se koriste kod LZ kodova (LZ77 i LZ78). Kod ovih postupaka polazi se od rječnika koji je na početku prazan ili određenog sadržaja. Rječnik posjeduje, grubo govoreći, dvije kolone. Jedna kolona se naziva prefiksom, koji predstavlja kôd neke fraze koja se već pojavljivala u poruci. U drugoj koloni se upisuje **sufiks**. Sufiks je bit ili simbol po kojem se nova fraza razlikuje od bilo koje koja se do tada pojavljivala u poruci. Kod LZ kodova rječnik se nikada ne prenosi, već se na strani prijema formira na isti način kako se formirao i na strani predaje. Pored optimalnosti (pod razmatranim uslovima), ovo je druga bitna prednost LZ kodnih postupaka. Ilustrujmo kodiranje i dekodiranje kod LZ kodova kroz dva primjera koji će dobrim dijelom pojasniti i pojednostaviti ovaj uvod.

Primjer III.11. Kodirajte binarnu poruku 1011010100010 putem LZ koda sa rječnikom koji je na početku prazan, sa 8 upisa koji se kodiraju putem trobitne binarne sekvence. Rječnik se po punjenu prazni. Prikazati proces dekodiranja.

Rješenje: Uslovom zadatka je rečeno da je rječnik na početku prazan (jedno polje smo zauzeli da bismo naznačili praznu frazu). Rječnik je prikazan u Tabeli III.5. U sekvenci tražimo dio koji se do sada nije pojavio u rječniku. To je već prvi bit. Kodiramo ga kao prefiks fraze prethodnika (to je ništa, ali je kod prefiksa 000) i sufiks po kome se nova fraza razlikuje od svega što je prethodilo (a to je bit 1). Dakle, šaljemo:

(000,1), a u rječnik smještamo frazu 1.

Zatim, u nekodiranom dijelu poruke pronalazimo ono što se do sada nije pojavljivalo u rječniku (to je zapravo drugi bit 0), pa informaciju o njemu šaljemo

kao kôd prefiksa (ponovo 000) i sufiks 0 (po kome se ova fraza razlikuje od svih dosadašnjih fraza):

(000, 0), a u rječnik smještamo frazu 0.

Sljedeći bit u nekodiranom dijelu sekvence je 1, ali on već postoji u rječniku, pa kodiramo kombinaciju trećeg i četvrtog bita, 11, kao prefiks 1 (kod prefiksa je 001) i sufiks (po kome se ova fraza razlikuje od prethodnih fraza). Dakle, šalje se:

(001, 1), a u rječnik se smješta 11.

Kombinacija 01 se kodira kao (010,1), a zatim se i ona smješta u rječnik. Sljedeća kombinacija/fraza u nekodiranom dijelu sekvence, koja se do sada nije pojavljivala u rječniku, jeste 010, koja se kodira kao (100,0) (jer je 100 kod fraze prefiksa 01). U rječnik se smješta 010. Naredna fraza 00 kodira se kao (010,0), Konačno, 10 se kodira sa (010,0). Poruka koja se šalje je:

(000,1)(000,0)(001,1)(010,1)(100,0)(010,0)(010,0).

Zagrade i znaci interpunkcije su, naravno, virtuelni i koristimo ih u primjeru samo da bismo vizuelno izdvojili djelove koda za pojedine fraze.

Objasnimo sada proces dekodiranja. Započinje istim rječnikom kao i proces kodiranja, odnosno od praznoga rječnika. Kada primimo (000,1), zaključujemo da je primljen bit sa prefiksnom „ništa“ (ponekada se, kao u programskim jezicima, navodi kao NULL) i 1, odnosno primili smo bit 1. Upisujemo ga u rječnik na poziciju kodiranu sa 001. Zatim primamo (000,0), dekodiramo 0 i upisujemo je u rječnik na poziciju kodiranu sa 010. Dalje primamo (001,1), to znači da primamo

Red. br.	Kod prefiksa	Fraza
0	000	-
1	001	1
2	010	0
3	011	11
4	100	01
5	101	010
6	110	00
7	111	10

Tabela III.5. Rječnik za LZ kod iz Primjera III.11

Red. broj	1	2	3	4	5	6	7	8	9	...
Fraza	0	1	2	01	012	10	101	00	000	...

Tabela III.6. Rječnik za LZ kod iz Primjera III.12

11 (prefiks je 1), a ovu frazu (11) upisujemo na poziciju koja je kodirana sa 011. Nastavak procesa možete provesti sami.□

Primjer III.12. Dat je ternarni alfabet. Rječnik na početku posjeduje kôd izvora. Fraze se kodiraju rednim brojevima, a rječnik ima beskonačan kapacitet. Kodirati poruku:

010121010100000...

Izvršiti kodiranje i dekodniranje LZ postupkom.

Rješenje: Činjenica da rječnik na početku nije prazan, već da posjeduje kod izvora, podrazumijeva da se na prve tri pozicije u rječniku upisuju elementi ulaznog alfabeta (ovdje 0, 1, 2), što je prikazano u Tabeli III.6. Ponovo kodiramo, počinjući sa početka koda, tražeći dio koji se do sada nije pojavio u rječniku, a to je kombinacija 01, gdje je 0 prefiks koji postoji u rječniku na poziciji 1 sa sufiksnom 1, po kojoj se ova fraza razlikuje od bilo koje koja do sada postoji u rječniku. Ovo se, dakle, kodira kao (1,2), a u rječnik se smješta kombinacija 01 na poziciju s rednim brojem 4. Ponavljamo postupak nad nekodiranim dijelom sekvence. Sada je sljedeća fraza koja se nije pojavljivala u rječniku 012, gdje je prefiks 01 (redni broj 4 u rječniku) sa sufiksnom 2 (u rječnik upisujemo 012 na poziciju sa rednim brojem 5). Nastavljamo postupak, kodirajući redom fraze 10, 101, 00, 000, što je propraćeno u rječniku.

Dakle, primljena poruka je:

(1,1)(4,2)(2,0)(6,1)(1,0)(8,0)...

Ni u ovom slučaju nema potrebe da se prema prijemniku prenosi rječnik. I prijemna strana počinje rječnikom koji ima kod izvora. Primamo (1,1), to znači da primamo simbol koji se nalazi na prvoj poziciji u rječniku (to je 0), sa sufiksom 1 (to je dekodirana fraza 01 koja se upisuje u rječnik na poziciju 4). Primamo zatim (4,2), a to je prethodna fraza 01 sa sufiksnom 2 (012 je dekodirano i upisuje se u rječnik). Nastavljamo sa (2,0), a to je prefiks 1 sa sufiksom 0 (dekodiramo 10 i to upisujemo u rječnik na poziciju 5 itd. □

Uočimo da nismo ostvarili kompresiju, posebno u prvom slučaju (iz Primjera III.11), jer je očigledno broj bita sada veći nego što je bio u ulaznoj poruci. Dakle, ostvarili smo negativnu kompresiju! Međutim, to nije toliki problem. LZ kod namijenjen je za duže sekvence, kada će postepeno da se (drastično) povećava dužina fraza koje se kodiraju i da se postepeno ostvaruje kompresija. Drugi problem koji se može uočiti je sama složenost procesa pretraživanja rječnika. Dakle, realizacija rječnika i procesa njegovog popunjavanja očigledno je od velike važnosti da bi brzina realizacije ovog kodnog postupka u praksi bila zadovoljavajuća. Međutim, ovo je ponovo samo tehničko pitanje.

Mnogo značajnija pitanja su ona koja se odnose na uticaj veličine rječnika na ostvarenu kompresiju, inicijalizaciju rječnika (početi od praznoga ili s nekim elementima), na postupak kada se rječnik napuni, adaptaciju (što raditi sa sekvencama koje se rijetko pojavljuju), kodiranje prefiksa itd. Obećavajući postupak, kakav je bio LZ kod, sa velikim brojem nepoznanica bio je predmet intenzivnog istraživanja u godinama koje su slijedile (krajem 1970-ih i početkom 1980-ih).

III.6.3 LZW algoritam

Velč je 1984. godine dao ključnu modifikaciju LZ koda koja se naziva LZW kodom. Predmetni kod je do danas najpopularniji za kompresiju bez gubitaka, koji se koristi kod mnogih programa za kompresiju fajlova na računarskim sistemima, ali i kod specijalizovanih programa za kompresiju multimedijalnih podataka ili za komunikacione potrebe.

Osnovna modifikacija je u izbjegavanju slanja sufiksa, odnosno uvijek se šalje samo šifra fraze iz rječnika (rječnik obično sadrži barem kod izvora, odnosno polazni alfabet). Šifre rječnika se postepeno povećavaju kako se rječnik puni i na taj način se izbjegava gubitak u kodiranju, koji se pojavljuje ako je šifra prefiksa (fraze) fiksne dužine (odnosno predugačka na početku).

Srećom, u stanju smo na jednom primjeru, relativno male dužine, ilustrovati kako je LZW kod efikasan i kako može na nekoliko desetina karaktera ostvariti određenu, nezanemarljivu uštedu (kompresiju). Naravno, primjer je specifičan jer u njemu postoji ponavljajuća struktura, ali je indikativan i za duže sekvence kod kojih ne postoji ovako očigledna ponavljajuća struktura.

Primjer III.13. Kodirati nabrajalicu iz našeg djetinjstva:

„ECI PECI PEC TI SI MALI ZEC A JA MALA PREPELICA ECI PECI PEC“

putem LZW koda. Rječnik ima 30 upisa za 30 glasova našeg jezika (LJ, NJ i DŽ ćemo brojati kao 1). Glasovi će biti kodirani binarnim ekvivalentom pozicije u azbuci datog slova (A = 1 = 00001, M = 15 = 01111, Š = 30 = 11110). Kombinaciju 00000 možemo koristiti da označi kraj sekvence, mada je u ovom primjeru ne koristimo, dok je bjelina kodirana kao _ = 11111 (označavamo je podvlakom da bismo je vizeuelno razlikovali od ostalih karaktera).

Rješenje: Rečenica (nabrajalica) se sastoji od 60 karaktera (47 slova i 13 bjelina). Dakle, potrebno nam je $60 \times 5 = 300$ bita za kodiranje ako svaki simbol kodiramo (a u ovom slučaju možemo) sa 5 bita. Kod LZW postupka kodira se najduži podskup (fraza) iz nekodiranog dijela sekvence koji postoji u rječniku, a u rječnik se smješta fraza koja ne postoji u rječniku (pohrani se za dalje kodiranje). Dakle,

šalje se samo kod faze prefiksa, a prefiks + sufiks se smješta u rječnik. Počnimo sa kodiranjem, uz objašnjavanje nekih ključnih koraka.

1. Šaljemo E (kodiramo kao 00111), a u rječnik smještamo EC i kodiramo kao 10000.

Dakle, ovo bi bio 32 upis u rječnik, pa ga moramo kodirati sa 6 bita, ali sve simbole nadalje moramo kodirati sa 6 bita, jer nemamo drugi način da ih razlikujemo od prethodnih sekvenci.

2. Šaljemo C (kod 011011), a u rječnik smještamo CI (kod 100001). Tekući gubitak 1 bit (koristimo šest, umjesto 5 bita).
3. Šaljemo I (kod 001010), a u rječnik smještamo I_ (kod 100010). Tekući gubitak 2 bita.
4. Šaljemo _ (000000), a u rječnik smještamo _P (kod 100011). Tekući gubitak 3 bita.
5. Šaljemo P (010011), a u rječnik smještamo PE (kod 100100), gubitak četiri bita.
6. Šaljemo EC (100000), a u rječnik smještamo ECI (100101), gubitak nula bita.

Dakle, sada smo u jednom potezu kodirali frazu od dva karaktera jer postoji u rječniku i umjesto sa 10 bita kodirali je sa šest bita i ostvarili dobit od četiri bita, čime smo poništili dosadašnji gubitak.

7. Šaljemo I_ (100010), u rječnik I_P (100110), dobitak četiri bita.
8. Šaljemo PE (100100), u rječnik PEC (100111), dobitak osam bita.
9. Šaljemo C (011011), u rječnik C_ (101000), dobitak sedam bita.
10. Šaljemo _ (011111), u rječnik _T (101001), dobitak šest bita.
11. Šaljemo T (010110), u rječnik TI (101010), dobitak pet bita.
12. Šaljemo I_ (100010), u rječnik I_S (101011), dobitak devet bita.
13. Šaljemo S (010101), u rječnik SI (101100), dobitak osam bita.
14. Šaljemo I_ (100010), u rječnik I_M (101101), dobitak 12 bita.
15. Šaljemo M (001111), u rječnik MA (101110), dobitak 11 bita.
16. Šaljemo A (000001), u rječnik AL (101111), dobitak 10 bita.
17. Šaljemo L (011101), u rječnik LI (110000), dobitak 9 bita.
18. Šaljemo I_ (100010), u rječnik I_Z (110001), dobitak 13 bita.
19. Šaljemo Z (001001), u rječnik ZE (110010), dobitak 12 bita.
20. Šaljemo EC (100000), u rječnik EC_ (110011), dobitak 16 bita.
21. Šaljemo _ (011111), u rječnik _A (110100), dobitak 15 bita.
22. Šaljemo A (000001), u rječnik A_ (110101), dobitak 14 bita.
23. Šaljemo _ (011111), u rječnik _J (110110), dobitak 13 bita.
24. Šaljemo J (001011), u rječnik JA (110111), dobitak 12 bita.

25. Šaljemo A_ (110101), u rječnik A_M (111000), dobitak 16 bita.
26. Šaljemo MA (101110), u rječnik MAL (111001), dobitak 20 bita.
27. Šaljemo L (001101), u rječnik LA (111010), dobitak 19 bita.
28. Šaljemo A_ (110101), u rječnik A_P (111011), dobitak 23 bita.
29. Šaljemo P (010011), u rječnik PR (111100), dobitak 22 bita.
30. Šaljemo R (010100), u rječnik RE (111101), dobitak 21 bit.
31. Šaljemo E (000111), u rječnik EP (111110), dobitak 20 bita.
32. Šaljemo PE (100100), u rječnik PEL (111111), dobitak 24 bita.
33. Šaljemo LI (10000), u rječnik LIC (1000000), dobitak 28 bita.

Kako je od ove tačke rječnik postao sedmobitni, sve fraze nadalje kodiramo sa sedam bita.

34. Šaljemo C (0011011), u rječnik CA (1000001), dobitak 26 bita.
35. Šaljemo A_ (0110101), u rječnik A_E (1000010), dobitak 29 bita.
36. Šaljemo ECI (0100101), u rječnik smještamo ECI_ (1000011), dobitak 37 bita.

Uočimo da smo po prvi put kodirali frazu od tri simbola, te da ušteda znatno raste.

37. Šaljemo _P (0100011), u rječnik smještamo _PE (1000100), dobitak 40 bita.
38. Šaljemo ECI_ (1000011), u rječnik smještamo ECI_P (1000101), dobitak 53 bita.

Sada smo sekvencu sa četiri bita kodirali odjednom sa sedam bita i ostvarili uštedu od 13 bita.

39. Šaljem PEC (01000111) čime ostvarujem ukupnu dobit od 61 bita.□

Dakle, u ovoj kratkoj sekvenci od 60 karaktera, ostvarili smo uštedu od 61 bita, to jest nešto preko 20%, što je veoma značajno. Naravno, ova ušteda je prouzrokovana dobrim dijelom činjenicom da je u pitanju veoma specifična poruka, ali dobro ukazuje na ponašanje LZW koda. Kako prolazimo kroz poruku, a u svakoj prirodnoj poruci će se pojaviti ponavljajući djelovi, dolazimo do postepenog povećanja dužine fraza čime se postiže sve veća i veća ušteda. Tipične ostvarene uštede idu oko 35% mada bitno zavise od strukture poruke.

Dekodiranje LZW koda se ipak obavlja s malom dozom opreza. Naime, ako pogledamo prethodni primjer i primimo prvi simbol, mi još uvijek ne znamo što je tim simbolom upisano u rječnik (znamo da je došlo do uvećanja rječnika, ali ne i ishod čitavog tog postupanja). Tek kada primimo naredni simbol znamo da je koder u prethodnom koraku u rječnik upisao, u našem slučaju, EC. Dakle, kod dekodiranja se proces upisivanja u rječnik obavlja sa jednim korakom zaostatka (nema uticaj na performanse dekodiranja), što je cijena za izbjegavanje slanja sufiksa.

LZW postupak je detaljno izučavan u literaturi i implementiran je u mnoštvu postupaka, tehničkih, hardeverskih, softverskih rješenja i standarda. Radi adekvatnih karakteristika (prije svega u smislu pretrage fraza), veličina rječnika se ograničava, a nekim od najpopularnijih standardnih postupaka maksimalni kod fraze rječnika je 12 bita, odnosno kod dozvoljava do 2^{12} upisa. Naravno, riječ je o karakteristikama koje mogu da se mijenjaju od tipa do tipa podataka i od primjene do primjene.

Od 2004. godine ovaj kodni postupak u osnovnoj formi nije zaštićen patentnim pravima i može se slobodno koristiti, što mu uvećava upotrebljivost.

III.7 Aritmetičko kodiranje

Aritmetičko kodiranje je naziv za više postupaka koji su alternativa Hafmenovom kodiranju. Neki od tih postupaka bitno se razlikuju po algoritamskoj osnovi i matematičkom aparatu. Ovdje ćemo nešto detaljnije proći kroz aritmetički kod koji se zasniva na određivanju pseudovjerovatnoće. Na kraju ćemo se kratko osvrnuti i na druge oblike aritmetičkog kodiranja.

Aritmetičko kodiranje zasnovano na pseudovjerovatnoći donekle je hibrid Hafmenovog i LZW kodiranja. Suštinski ne zahtijeva poznavanje vjerovatnoća, ni uslovnih vjerovatnoća među simbolima. Takođe ne zahtijeva prenošenje rječnika sa strane predaje na stranu prijema. Po performansama blizak je postupak Hafmenovom kodiranju. Dugo vremena bio je ograničene primjenljivosti pošto je nerado korišćen zbog pripadajućih autorskih prava. U periodu 2010–2015. godine istekla je većina patentnih prava na aritmetičko kodiranje, pa se može očekivati da ovaj postupak dobije na popularnosti, jer se sada većina algoritama može koristiti bez plaćanja autorskih prava.

Aritmetičko kodiranje ne formira rječnik, već poruku koju prima pretvara u **pseudovjerovatnoću**. Pseudovjerovatnoća samo podsjeća na pravu vjerovatnoću (nalazi se u granicama od 0 do 1), ali nije prava vrijednost vjerovatnoće neke sekvence koja se prenosi. Nakon što se odredi pseudovjerovatnoća (ili tokom njenog određivanja), vrši se pretvaranje u poruku koja se prenosi.

Pretpostavimo da imamo alfabet X koji se sastoji od simbola $\{x_1, x_2, \ldots, x_N\}$ (ukupno N simbola). Pored toga, imamo dodatni karakter koji ćemo označiti sa # i koji nazivamo **terminacionim** i koji predstavlja kraj poruke. Pretpostavimo da na početku znamo početne vjerovatnoće simbola alfabeta $\{p_1^{(0)}, p_2^{(0)}, \ldots, p_N^{(0)}\}$ i terminacionog simbola $p_{\#}^{(0)}$. Očigledno mora da važi:

$$\sum_{i=1}^{N} p_i^{(0)} + p_{\#}^{(0)} = 1.$$

Sekvenca koju ćemo poslati može da ima pridruženu bilo koju vjerovatnoću iz intervala [0,1). Početna vrijednost intervala označava se obično sa $u=0$, krajnja $v=1$, dok se širina intervala obično označava sa $p=v-u$ i na početku iznosi $p=1$. Na početku možemo da pošaljemo bilo koji karakter iz alfabeta X ili terminacioni karakter # i svakom od tih karaktera pridružujemo dio proporcionalan početnoj vjerovatnoći na brojnoj duži (Slika III.11). Donje i gornje granice pojedinih intervala su (za simbole iz alfabeta):

$$\text{za } x_i \text{ donja granica } u_i^{(0)} = u + \sum_{j=0}^{i-1} p_j^{(0)}, \text{ gornja granica } v_i^{(0)} = u + \sum_{j=0}^{i} p_j^{(0)},$$

dok je za terminacioni simbol:

$$\text{donja granica } u_{\#}^{(0)} = u + \sum_{j=0}^{N} p_j^{(0)} = 1 - p_{\#}^{(0)}, \text{ gornja granica } v_{\$}^{(0)} = 1.$$

Kada izaberemo l-ti simbol x_l iz alfabeta, zapravo smo sveli zonu mogućih vjerovatnoća na interval (Slika III.11):

$$[u_l^{(0)}, v_l^{(0)}] = \left[u + \sum_{j=0}^{l-1} p_j^{(0)}, u + \sum_{j=0}^{l} p_j^{(0)} \right].$$

Širina predmetnog inervala je:

$$v_l^{(0)} - u_l^{(0)} = p_l^{(0)}.$$

Ovo se obično zapisuje tako da se ažuriraju vrijednosti granica u i v i širine intervala p:

$$u = u_l^{(0)} \quad v = v_l^{(0)} \quad p = v - u.$$

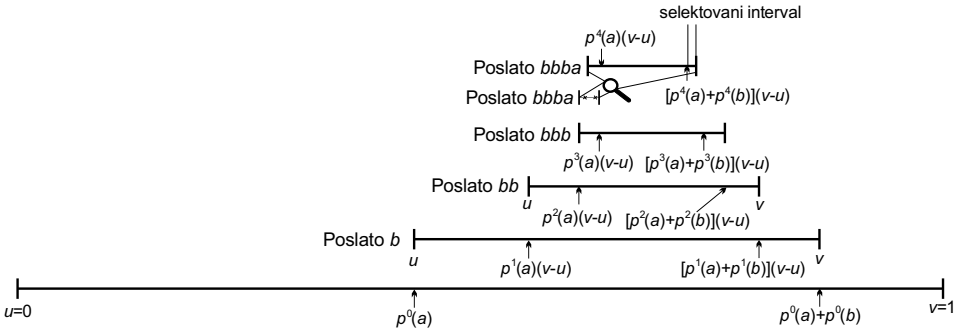
Ovo radimo da bi smanjili broj oznaka koje se koriste u algoritmu. Sada se dobijena duž podijeli na podintervale proporcionalne vjerovatnoćama $\{p_1^{(1)}, p_2^{(1)}, ..., p_N^{(1)}\}$, sa vjerovatnoćom terminacionog karaktera $p_{\#}^{(1)}$. Predmetna duž je sada podijeljena u podintervale, čije su početne i krajnje tačke, redom:

$$u_i^{(1)} = u + p\sum_{j=0}^{i-1} p_j^{(1)} \qquad v_i^{(1)} = u + p\sum_{j=0}^{i} p_j^{(1)} \qquad i \in [1, N]$$

$$u_{\#}^{(1)} = v - pp_{\#}^{(1)} \qquad v_{\#}^{(1)} = v.$$

Uočimo da su vjerovatnoće sada pomnožene sa širinom intervala (duži) kojem pripada selektovani dio. Ako pošaljemo simbol x_k, odabrali smo podinterval:

$$[u_k^{(1)}, v_k^{(1)}] = \left[u + p\sum_{j=0}^{k-1} p_j^{(0)}, u + p\sum_{j=0}^{k} p_j^{(0)} \right].$$



Slika III.11. Ilustracija računanja pseudovjerovatnoće kod aritmetičkog koda (u posljednjem koraku izvršeno je zumiranje selektovanog intervala)

Ponovo ažuriramo početnu i krajnju tačku i širinu intervala (Slika III.11), pa dobijamo:

$$u = u_k^{(1)} \quad v = v_k^{(1)} \quad p = v - u\,.$$

U svakom koraku postepeno sužavamo interval pseudovjerovatnoća koji korespondira sekvenci koju smo poslali. U *r*-tom koraku, granice intervala se mogu zapisati kao:

$$u_i^{(r)} = u + p\sum_{j=0}^{i-1} p_j^{(1)} \qquad v_i^{(r)} = u + p\sum_{j=0}^{i} p_j^{(1)} \qquad i \in [1, N]$$

$$u_{\#}^{(r)} = v - pp_{\#}^{(r)} \qquad v_{\#}^{(r)} = v.$$

Pretpostavimo da se u tom koraku šalje terminacioni karakter, pa je konačni selektovani interval pseudovjerovatnoća:

$$[u_{\#}^{(r)}, v_{\#}^{(r)}).$$

Pošto se radi o brojevima u intervalu od [0,1), možemo izvršiti konvertovanje granica intervala zapisanih u obliku dekadnog realnog broja u binarni. Podsjetimo se te operacije. Pretpostavimo da je vjerovatnoća jedne od granica 0.73. Pretvaranje se vrši množenjem sa dva, pamćenjem cjelobrojnog dijela, a zatim se decimalni dio ponovo množi sa dva:

$0.73 \times 2 = 1.46$ 0.1.....
$0.46 \times 2 = 0.92$ 0.10.....
$0.92 \times 2 = 1.84$ 0.101....
$0.84 \times 2 = 1.68$ 0.1011...
$0.68 \times 2 = 1.36$ 0.10111..
$0.36 \times 2 = 0.72$ 0.101110...

Procedura se može nastaviti dalje. Uočljivo je, naravno, da dobijamo broj koji ima konačan broj decimala samo u slučaju kada se vjerovatnoća može napisati u obliku $r/2^q$, gdje su r i q nenegativni cijeli brojevi. U svim ostalim slučajevima dobićemo „iracionalan“ binarni broj sa beskonačno decimala. Postavlja se pitanje: kako na osnovu binarnih sekvenci dobijenih putem granica intervala pseudovjerovatnoća $[u_{\#}^{(r)}, v_{\#}^{(r)})$ da dobijemo odgovarajući kod? Cilj procesa je da se postavi vjerovatnoća koja jedinstveno identifikuje navedeni interval i koja je što je moguće kraća, kako bi se uštedjelo na kodiranju (mada kod dugačkih sekvenci ovo ne mora biti od ključne važnosti). Suštinski, potrebno je da se odabere najkraća sekvenca koja zasigurno pripada intervalu posljednjeg (terminacionog) simbola. U našim primjerima mi se držimo navedenog pravila koje nije optimalno sa stanovišta odgovarajuće dužine kodne riječi. Za praktičnu implementaciju potrebno je izvršiti neki od postupaka koji ograničavaju dužinu kodne riječi, od kojih je najpopularniji renormalizacija. U suštini, tražimo broj oblika $r/2^q$ s najmanjim brojem decimalnih cifara q koji zasigurno pripada intervalu $[u_{\#}^{(r)}, v_{\#}^{(r)})$.

III.7.1 Algoritam

Zbog značaja i složenosti ovog algoritma, dajmo formalni prikaz algoritma kojim se vrši aritmetičko kodiranje.

Ulaz. Sekvenca simbola $\{y_1, y_2, \ldots, y_M, y_{M+1}\}$ gdje je posljednji simbol terminacioni $y_{M+1} = \#$. Simboli $\{y_1, y_2, \ldots, y_M\}$ izabrani su iz skupa X koji se sastoji od N simbola.

Inicijalizacija. Granice intervala se postavljaju na $u = 0$, $v = 1$, širina intervala je $p = v - u = 1$. Predmetni interval je podijeljen u $N+1$ podintervala, čije su granice:

$$[u_l^{(0)}, v_l^{(0)}] = \left[u + p\sum_{j=1}^{l-1} p_j^{(0)}, u + p\sum_{j=1}^{l} p_j^{(0)} \right], \quad l \in [1, N],$$

koje korespondiraju simbolima alfabeta X, dok interval:

$$\left[u + p\sum_{j=1}^{N} p_j^{(0)}, v \right]$$

odgovara terminacionom karakteru.

Ciklus: Za svako $i \in [1, M]$, u zavisnosti od toga kojem simbolu alfabeta X odgovara poslati karakter y_i, biramo interval:

$$[u_{l_i}^{(i)}, v_{l_i}^{(i)}] = \left[u + p\sum_{j=1}^{l_i-1} p_j^{(i-1)}, u + p\sum_{j=1}^{l_i} p_j^{(i-1)} \right],$$

pod pretpostavkom da je $y_i = x_{l_i}$. U slučaju selekcije terminacionog simbola, interval je:

$$[u_{\#}^{(i)}, v_{\#}^{(i)}] = \left[u + p\sum_{j=1}^{N} p_j^{(i-1)}, v \right].$$

Ažuriramo granice i širinu intervala:

$$u = u_{l_i}^{(i)} \quad v = v_{l_i}^{(i)} \quad p = v - u.$$

Po odgovarajućem pravilu, ažuriramo vjerovatnoće

$$p_j^{(i)}, \ j \in [1, N],$$

dok se vjerovatnoća terminacionog simbola $p_{\#}^{(i)}$ ažurira po alternativnom pravilu.

Kraj ciklusa

Posljednji simbol je terminacioni, kojem korespondira interval pseudovjerovatnoća:

$$[u_{\#}^{(M)}, v_{\#}^{(M)}] = \left[u + p\sum_{j=1}^{N} p_j^{(M-1)}, v \right].$$

Pretvore se granice intervala u binarne decimalne brojeve, a kao poruka se pošalje onaj decimalni dio binarnog zapisa broja koji zasigurno korespondira tom intervalu.

III.7.2 Pravila ažuriranja vjerovatnoća

U praksi se koriste dva pravila ažuriranja vjerovatnoća: Laplasovo i Dirikleovo (njem. *Dirichlet*). Prvo ima utemeljenje u teorijskim postavkama i Bajeosovom pravilu. Biće djelimično objašnjeno u Poglavlju III.7.4. Zbog odstupanja koja postoje u odnosu na teorijske postavke, u praksi se često koristi drugo pravilo. Objasnimo funkcionisanje ovih pravila.

Pretpostavimo da su početne vjerovatnoće svih simbola alfabeta jednake:

$$p_j^{(0)} = \frac{1 - p_{\#}^{(0)}}{N}.$$

Pretpostavimo dalje da je prvi poslati simbol x_1. Dakle, u tekućoj poruci imamo jedan simbol x_1, a svi ostali simboli nisu upotrijebljeni. Možemo prirodno pretpostaviti da se simbol x_1 pojavljuje češće od ostalih, a da ostali imaju jednaku vjerovatnoću (jer o njima još nemamo nikakvu informaciju). Pretpostavimo dalje da je ažurirana vjerovatnoća terminacionog simbola # $p_{\#}^{(1)}$, a to znači da je za simbole alfabeta X preostala vjerovatnoća $1 - p_{\#}^{(1)}$. Sada pretpostavimo da smo poslali određeni broj simbola alfabeta (neka je to L), te da smo brojali koliko puta se koji

od simbola $\{x_1, x_2, \ldots, x_N\}$ pojavio u sekvenci. Neka je broj pojavljivanja simbola $\{F_1, F_2, \ldots, F_N\}$. Po Laplasovom pravilu ažuriranja, pseudovjerovatnoće se određuju kao:

$$p_j^{(L)} = [1 - p_{\#}^{(0)}] \frac{F_j + 1}{\sum_{i=1}^{N} (F_i + 1)} = [1 - p_{\#}^{(0)}] \frac{F_j + 1}{\sum_{i=1}^{N} F_i + \sum_{i=1}^{N} 1} = [1 - p_{\#}^{(0)}] \frac{F_j + 1}{L + N}.$$

Dakle, što se simbol više puta (veće F_j) pojavio u sekvenci do sada, to je vjerovatnije da će se pojaviti i u ostatku poruke. Ovdje svim simbolima uvodimo sabirak 1 da ne bismo sveli na nulu pseudovjerovatnoću simbola koji se nisu pojavili u sekvenci. Kasnije ćemo pokazati da je pod određenim okolnostima ovakav izbor optimalan.

Drugi tip pravila ažuriranja naziva se Dirikleovim i po njemu se pseudovjerovatnoće ažuriraju na sljedeći način:

$$p_j^{(L)} = [1 - p_{\#}^{(0)}] \frac{F_j + \delta}{\sum_{i=1}^{N} (F_i + \delta)} = [1 - p_{\#}^{(0)}] \frac{F_j + \delta}{\sum_{i=1}^{N} F_i + \sum_{i=1}^{N} \delta} = [1 - p_{\#}^{(0)}] \frac{F_j + \delta}{L + N\delta},$$

gdje je δ mali broj, tipično manji od 0.1, a često se usvaja u granicama [0.01,0.05]. U praksi se ovakvo pravilo ažuriranja ponekad pokazuje boljim jer dobro simulira situaciju da se pojedini simboli iz alfabeta jako rijetko pojavljuju u sekvencama.

III.7.3 Primjer

Ilustrujmo predmetni kod putem jednog primjera kako bismo pojasnili način na koji se s ovim kodom radi.

Primjer III.14. Pretpostavimo da se alfabet sastoji od dva simbola $X = \{a,b\}$. Terminacioni simbol je #. Vjerovatnoća terminacionog simbola je 0.15 i ne mijenja se u poruci (u datom primjeru to je pominjano posebno pravilo ažuriranja terminacionog simbola). Na početku pretpostaviti da je vjerovatnoća simbola alfabeta X ista, te da se za ažuriranje pseudovjerovatnoća primjenjuje Laplasovo pravilo ažuriranja. Kodirati poruku:

bbba#.

Rješenje: Na početku, interval je $[u,v) = [0,1)$ širine $p = 1$. Početne vrijednosti vjerovatnoća su $p^{(0)}(a) = p^{(0)}(b) = (1 - 0.15)/2 = 0.425$, odnosno $p_{(0)}(\#) = 0.15$. Ovo odgovara intervalima: [0,0.425), [0.425,0.85), [0.85,1), respektivno. Kako šaljemo *b*, to znači da smo po slanju prvog simbola selektovali interval:

I simbol: $[u,v) = [0.425,0.85)$ širine $p = v - u = 0.425$.

Sada vršimo ažuriranje vjerovatnoća. Uočimo da je $F_a = 0$, a $F_b = 1$, odnosno da je do sada u poruci bio jedan simbol *b* i nijedan simbol *a*:

$$p^{(1)}(a) = [1 - p_{\#}^{(1)}]\frac{F_a + 1}{F_a + 1 + F_b + 1} = \frac{1 - 0.15}{3} = \frac{0.85}{3}$$

$$p^{(1)}(b) = [1 - p_{\#}^{(1)}]\frac{F_b + 1}{F_a + 1 + F_b + 1} = [1 - 0.15]\frac{2}{3} = \frac{2}{3}0.85.$$

Simbolima *a*, *b* i # sada odgovaraju intervali:

a: $[0.425, 0.425 + 0.425 \times 0.85/3) \approx [0.425, 0.5454)$
b: $\approx [0.5454,\ 0.425 + 0.425 \times 0.85) \approx [0.5454, 0.7862)$
#: $\approx [0.7862, 0.85)$.

Pošto šaljemo simbol *b*, to znači da smo odabrali interval:

II simbol $[u,v) \approx [0.5454, 0.7862)$ širine $p = v - u \approx 0.2408$.

Izvršimo ažuriranje pseudovjerovatnoća, imajući na umu da je $F_a = 0$ i $F_b = 2$:

$$p^{(2)}(a) = [1 - p_{\#}^{(2)}]\frac{F_a + 1}{F_a + 1 + F_b + 1} = 0.85\frac{1}{4} = 0.2125$$

$$p^{(2)}(b) = [1 - p_{\#}^{(2)}]\frac{F_b + 1}{F_a + 1 + F_b + 1} = 0.85\frac{3}{4} = 0.6375$$

sa asociranim intervalima:

a: $\approx [0.5454, 0.5454 \times 0.2125 \times 0.2408) \approx\ [0.5454, 0.5966)$
b: $\approx [0.5966, 0.7501)$
#: $\approx [0.7501, 0.7862)$.

Ponovo se šalje *b*, što znači da je izabran interval:

III simbol: $[u,v) \approx [0.5966, 0.7501)\ \ p \approx 0.1535$.

Nanovo ažuriramo pseudovjerovatnoće sa $F_a = 0$ i $F_b = 3$:

$$p^{(3)}(a) = [1 - p_{\#}^{(3)}]\frac{F_a + 1}{F_a + 1 + F_b + 1} = 0.17$$

$$p^{(3)}(b) = [1 - p_{\#}^{(3)}]\frac{F_b + 1}{F_a + 1 + F_b + 1} = 0.68$$

sa asociranim intervalima:

a: $\approx [0.5966, 0.6227)$
b: $\approx [0.6227, 0.7271)$
#: $\approx [0.7271, 0.7501)$.

Pošto je poslato a, svedeni smo na interval:

IV simbol: $[u,v) \approx [0.5966, 0.6227)$ $p \approx 0.0261$.

Sada raste vjerovatnoća izbora simbola a jer imamo situaciju da je $F_a = 1$ i $F_b = 3$:

$$p^{(4)}(a) = [1 - p_{\#}^{(4)}] \frac{F_a + 1}{F_a + 1 + F_b + 1} = \frac{0.85}{3} \quad p^{(4)}(b) = [1 - p_{\#}^{(4)}] \frac{F_b + 1}{F_a + 1 + F_b + 1} = \frac{1.7}{3}$$

sa asociranim intervalima:

a: $\approx [0.5966, 0.6040)$
b: $\approx [0.6040, 0.6188)$
#: $= [0.618779015625, 0.6226940625)$.

Kako se sada šalje terminacioni simbol, podrazumijeva se da smo odabrali interval pseudovjerovatnoća (sada zapisan do potpune tačnosti)

$$[0.618779015625, 0.6226940625).$$

Izvršimo pretvaranje granica intervala u binarni brojni zapis:

$0.618779015625 \times 2 =$	$1 + \approx 0.2375$	$0.6226940625 \times 2 =$	$1 + \approx 0.2454$
$0.2375 \times 2 =$	$0 + \approx 0.4751$	$0.2454 \times 2 =$	$0 + \approx 0.4908$
$0.4751 \times 2 =$	$0 + \approx 0.9502$	$0.4908 \times 2 =$	$0 + \approx 0.9816$
$0.9502 \times 2 =$	$1 + \approx 0.9005$	$0.9816 \times 2 =$	$1 + \approx 0.9632$
$0.9005 \times 2 =$	$1 + \approx 0.8009$	$0.9632 \times 2 =$	$1 + \approx 0.9262$
$0.8009 \times 2 =$	$1 + \approx 0.6019$	$0.9262 \times 2 =$	$1 + \approx 0.8524$
$0.6019 \times 2 =$	$1 + \approx 0.2037$	$0.8524 \times 2 =$	$1 + \approx 0.7048$
$0.2037 \times 2 =$	$0 + \approx 0.4074$	0.7048×2	$1 + \approx 0.4097$
$0.4074 \times 2 =$	$0 + \approx 0.8149$	0.4097×2	$0 + \approx 0.8194$.

...

Vidimo da su granice intervala, predstavljene u binarnom zapisu, jednake na prvih sedam decimalnih mjesta 1001111. Nakon toga, počinju se razlikovati, odnosno naredna dva bita za početak intervala su 00, dok su naredna dva bita za kraj intervala 10. Ovo, na primjer, znači da je bezbjedno kodirati interval dodajući dva bita 01, jer smo sigurni da ta tačka pripada intervalu. Ovakav postupak kod kraćih poruka može da bude nepraktičan jer dodaje nekoliko bita na kraj poruke. Kod realističnih poruka, koje su mnogo duže nego što je ova u primjeru, dodati nekoliko bita na kraj intervala nije problematično. Naime, u praktičnim implementacijama poruke se često odvajaju sekvencom od nekoliko bita, koja predstavlja terminaciju čitave poruke. Dakle, poruka se u posmatranom primjeru mogla kodirati kao:

$$100111101,$$

odnosno sa 9 bita ukupno (ovo korespondira sa vjerovatnoćom pominjanog oblika $r/2^q$, gdje je u ovom slučaju $r = 317$, dok je $q = 9$, odnosno 317/512).

Postavlja se pitanje: da li je potrebno čekati kraj poruke i određivanje konačne pseudovjerovatnoće, odnosno granica intervala da bi se poruka kodirala? Nema potrebe napominjati da bi ovo značilo veliko kašnjenje u slanju poruke i da takav postupak ne bi bio praktično primjenljiv. Naravno, to nije slučaj.

U prvom koraku naš selektovani interval bio je:

$$[u,v) = [0.425,0.85).$$

Pretvaranjem granica intervala u binarne brojeve dobijamo: [0.0110…, 0.1101…), dakle ne postoji sigurnost u prvu binarnu cifru, koja je zajednička za granice intervala, pa ne možemo poslati nijedan bit u kanal. Nakon drugog simbola imamo granice intervala:

$$[0.5454,0.7862).$$

Binarno predstavljane granice intervala su: [0.1000…, 0.1100…), tako da smo sigurni u prvi bit koji je zajednički za granice intervala, pa bit 1 puštamo u kanal. Nakon trećeg simbola, interval pseudovjerovatnoća je ograničen na:

$$[0.5966,0.7501).$$

Međutim, ponovo nismo sigurni u sadržaj drugog bita, pa ga ne možemo poslati u kanal. Naime, 0.75 je vjerovatnoća koja dijeli binarno zapisani podinterval [0.10…, 0.11…), a naš interval je s obje strane ove granice. Poslije četvrtog simbola imamo:

$$[0.5966,0.6227),$$

što se pretvaranjem u binarne brojeve može prikazati kao:

$$[0.10010\ldots, 0.10011\ldots).$$

Dakle, sada smo sigurni u drugi, treći i četvrti bit, pa šaljemo u kanal 101. Preostale bite poruke šaljemo nakon terminacionog simbola. Očigledno je da predmetnom metodologijom ne možemo obezbijediti da biti pristižu u kanal na isti način kako pristižu u koder. Ipak, u standardnom prenosu česti su baferi ili paketi koji prikupljaju veći broj bita i šalju ih zajedno, pa predmetno kašnjenje nije od veće važnosti.

Preostaje da se odgovori na pitanje da li je potrebno u procesu dekodiranja primiti čitavu poruku ili se proces dekodiranja može vršiti tokom prijema pojedinih bitova. Ponovo se proces dekodiranja može vršiti tokom prijema bitova, istina, ne uniformnom brzinom, što ćemo demonstrirati na primjeru našeg koda. Dekodiranje počinje od tri inicijalna intervala:

$$[0,0.425), [0.425,0.85), [0.85,1).$$

Kako je primljena poruka 100111101, odnosno kako je prvi primljeni bit 1, to znači da je poruka u zoni vjerovatnoća $[0.1,0.1111\ldots)_2$ u binarnom brojnom sistemu, odnosno $[0.5,1)_{10}$ u dekadnom. Vidimo da dva intervala imaju presjek s ovim intervalom, a to su intervali koji korespondiraju simbolu *b* i terminacionom simbolu. Dakle, prijemom prvog bita poruke, ne možemo izvršiti dekodiranje. Prijemom drugog simbola poruke 0, svodimo interval na granice $[0.10,0.10111\ldots)_2$, odnosno $[0.5,0.75)_{10}$, pa smo sigurni da je primljeni simbol *b*, vršimo dekodiranje i proračun pseudovjerovatnoća sa tri moguća podintervala:

$$[0.425,0.5454), [0.5454,0.7862), [0.7862,0.85).$$

Naredni primljeni bit je 0, pa imamo vjerovatnoće u granicama $[0.100,0.100111\ldots)_2$, odnosno $[0.5,0.625)_{10}$, a ne možemo da izvršimo dekodiranje jer ovaj interval ima presjeke s intervalima koji odgovaraju simbolima *a* i *b*. Nakon toga primamo bit 1 što svodi naše vjerovatnoće u granice $[0.1001,0.100111\ldots)_2$, odnosno $[0.5625,0.625)_{10}$, čime smo sigurnu da je drugi primljeni simbol *b*, vršimo korespondentno dekodiranje, pa rekalkulišemo vjerovatnoće (na isti način kao kod kodiranja), nakon čega dobijamo sljedeće intervale za simbole alfabeta i terminacioni simbol:

$$[0.5454,0.5966), [0.5966,0.7501), [0.7501,0.7862).$$

Naredni bit poruke koju primamo ponovo je 1, čime se interval svodi na $[0.10011,0.100111\ldots)_2$, odnosno $[0.59375,0.625)_{10}$, ali pošto ovaj interval ima presjeke s intervalima koji korespondiraju simbolima *a* i *b* i dalje imamo neodređenost u prijemu trećeg simbola. Naredni primljeni bit (ponovo jedinica) sužava interval na $[0.100111,0.100111111\ldots)_2$, odnosno $[0.609375,0.625)_{10}$, čime potvrđujemo da je treći primljeni simbol ponovo *b*, vršimo ponovno računanje vjerovatnoća i svodimo intervale korespondentne simbolima alfabeta i terminacionom simbolu na:

$$[0.5966,0.6040), [0.6040,0.6188), [0.6188,0.6227).$$

Ostavljamo čitaocima da, za vježbu, provedu do kraja predmetni postupak.

*III.7.4 Model vjerovatnoće kod aritmetičkog kodiranja**

Kvalitet aritmetičkog kodiranja u velikoj mjeri zavisi od modela ažuriranja vjerovatnoća. Posmatrajmo problem opisan Primjerom III.14 sa vjerovatnoćama simbola $p(a)$, $p(b)$ i $p(\#)$ koje obično nisu poznate unaprijed, a često ne postoji ni neka procjena koliko će u partikularnoj poruci biti pojedinih simbola. Ako pretpostavke postoje, one se mogu ugraditi u proces aritmetičkog kodiranja i takvi postupci nisu rijetki u praksi. Kada nemamo pretpostavke o vjerovatnoći nekog simbola, možemo je modelovati nekom funkcijom gustine raspodjele, a relativno čest model

je uniformna raspodjela. Pretpostavimo da imamo binarni alfabet kakav je bio u Primjeru III.14. Pretpostavimo da smo primili F simbola od kojih je F_a simbola a i F_b simbola b ($F_a + F_b = F$). Vjerovatnoća da se simbol a pojavljuje sa vjerovatnoćom p_a nakon predmetne sekvence, može se zapisati:

$$P(p_a \mid s, F) = \frac{P(s \mid p_a, F) P(p_a)}{P(s \mid F)},$$

gdje je s konkretna primljena sekvenca simbola (bita) a i b. Uslovna vjerovatnoća u brojiocu može se zapisati kao:

$$P(s \mid p_a, F) = p_a^{F_a} p_b^{F_b} = p_a^{F_a} (1 - p_a)^{F_b},$$

gdje smo uzeli $p_b = 1 - p_a$ i znamo da sekvenca još nije završena, odnosno da se u njoj nije mogao pojaviti terminacioni simbol. Dalje se može odrediti:

$$P(s \mid F) = \int_0^1 P(s \mid p_a, F) P(p_a) dp_a = \int_0^1 p_a^{F_a} (1 - p_a)^{F_b} P(p_a) dp_a.$$

Kako nemamo informacija o vjerovatnoći simbola, možemo da smatramo da je uniformna na intervalu [0,1) (pod uslovom da se nije mogao pojaviti terminacioni simbol), pa slijedi:

$$P(s \mid F) = \int_0^1 p_a^{F_a} (1 - p_a)^{F_b} dp_a.$$

Ovaj integral se može izraziti u obliku specijalne beta funkcije $B(x,y)$, kao:

$$P(s \mid F) = \int_0^1 p_a^{F_a} (1 - p_a)^{F_b} dp_a = B(F_a + 1, F_b + 1).$$

Ova funkcija se može izraziti preko gama funkcije $G(x)$, kao $B(x,y) = G(x)G(y)/G(x+y)$, odnosno u našem slučaju:

$$B(F_a + 1, F_b + 1) = \frac{G(F_a + 1) G(F_b + 1)}{G(F_a + F_b + 2)}.$$

Srećom, gama funkcija se za cjelobrojne argumente svodi na prosti faktorijel, pa imamo:

$$P(s \mid F) = \frac{(F_a + 1)!(F_b + 1)!}{(F_a + F_b + 2)!}.$$

Sada se može odrediti vjerovatnoća p_a koja maksimizuje vjerovatnoću $P(p_a \mid s, F)$. Na primjer za $P(p_a \mid s = aba, F = 3) \propto p_a^2 (1 - p_a)$ što se maksimizuje za $p_a = 2/3$, što se lako pokazuje diferenciranjem po p_a i izjednačavanjem s nulom

$[p_a^2(1-p_a)]' = 2p_a - 3p_a^2 = 0 \Rightarrow p_a = 2/3$. Takođe, željeli bismo da smo u stanju da vršimo predikciju. Ako je data sekvenca s dužine F i potrebno je da odredimo vjerovatnoću (predikciju) simbola a, slijedi:

$$P(a \mid s,F) = \int_0^1 P(a \mid p_a) P(p_a \mid s,F) dp_a .$$

U ovom slučaju, unosimo kompletnu neodređenost o p_a. Na osnovu ovoga može se odrediti prediktor:

$$P(a \mid s,F) = \int_0^1 p_a \frac{p_a^{F_a}(1-p_a)^{F_a}}{P(s \mid F)} dp_a = \frac{F_a + 1}{F_a + F_b + 2}.$$

Dobili smo Laplasovo pravilo ažuriranja za binarni alfabet, pod pretpostavkom da se u sekvenci date dužine nije mogao pojaviti terminacioni simbol. Generalizacijom za više simbola u alfabetu dobijamo uvedeno pravilo ažuriranja, uz posebno pravilo za terminacioni simbol. Treba imati na umu da optimalnost Laplasovog pravila ažuriranja potiče iz uvedene pretpostavke da je vjerovatnoća simbola uniformno raspodijeljena. To u praksi često nije slučaj. Pojedini simboli alfabeta mogu da se pojave rijetko, pa se stoga ponekad koristi Dirikleovo pravilo ažuriranja koje može da pokaže nešto bolje rezultate nego Laplasovo pravilo.

III.7.5 Aritmetičko kodiranje sa korelacijom*

Ponekad se u literaturi, pa i praksi, najčešće u multimedijima, aritmetičkim kodiranjem naziva postupak koji bitno odstupa od prethodnog. U prethodnom postupku vršili smo određivanje pseudovjerovatnoće pojave određene sekvence među svim sekvencama i na odgovarajući način kodiranu informaciju o sekvenci prosljeđivali u kanal. Kod pseudovjerovatnoća uspostavljamo statističku vezu između dijela sekvence koji je bio sa tekućim simbolom. Postoji i drugi način da se izvrši opisivanje statističkih veza između pojedinih simbola. Algebarski možemo to opisati kao:

$$x(n+1) = a_0 x(n) + a_1 x(n-1) + \ldots + a_Q x(n-Q) + \nu(n+1) =$$

$$= \sum_{i=0}^{Q} a_i x(n-i) + \nu(n+1).$$

Predmetni model se često naziva **autoregresivni**. Ovdje su a_i koeficijenti, dok bi $\nu(n)$ trebalo biti mali ostatak (rezidualna vrijednost) iz ovakve procedure. Ostatak $\nu(n)$ se obično može modelovati kao šum – slučajni proces. Vrlina ovakvog procesa kodiranja bila bi u tome što je neophodno zapamtiti mali broj koeficijenata a_i, a zatim kodirati ostatak $\nu(n)$ nekom efikasnom procedurom. Predmetna tehnika pretpostavlja da je moguće dobiti relativno mali ostatak koji se može efikasno

kodirati, sa velikim stepenom kompresije. Alternativno se, poništavanjem malih vrijednosti, nad ostatkom može provesti procedura kompresije sa gubicima.

Osnovni problem ovakvog postupka je u određivanju koeficijenata a_i, te u određivanju dužine Q sa kojom je moguće obaviti predmetni postupak na kvalitetan način. Najveći broj algoritama zasnovan je na određivanju autokorelacije signala, a što posljedično povlači matrični račun, određivanje sopstvenih ili singuralnih vrijednosti itd. Dakle, veliki problem je složenost i neizvjesnost ovakvih postupaka, te se, posebno u komunikacijama, rijetko provode.

Uočimo da je ovakav postupak neka vrsta generalizacije diferencijalnog kodiranja. Kod diferencijalnog kodiranja usvajamo da je „šum“ razlika susjednih odbiraka, a takav postupak, kao što smo već rekli, opravdan je u nekim primjenama. Stoga, njegova generalizacija putem autoregresivnog modela ima određeni značaj u pojedinim oblastima tehnike, a već smo pomenuli multimedijalne signale (posebno slike), gdje se susjedni odbirci signala (kod slike to su pikseli) malo razlikuju.

III.8 Zadaci i softverska realizacija

III.8.1 Riješeni zadaci

3.1. Posmatran je fajl s podacima na engleskom jeziku. Broj pojavljivanja određenih simbola dat je u Tabeli III.7. Odrediti tipičnu sekvencu za predmetni model, ako se posmatra 100 karaktera teksta.

Rješenje: Pojava pojedinih karaktera data je u Tabeli III.8 u procentima. U tipičnoj sekvenci od 100 simbola *a* se pojavljuje barem 8 puta, *b* jednom, *c* tri puta, *d* četiri puta, *e* dvanaest puta, *f* dva puta, *g* jednom, *h* četiri puta, *i* sedam puta, *l* četiri

a	4186	*b*	832	*c*	1589	*d*	2045	*e*	6103	*f*	1050
g	947	*h*	2232	*i*	3785	*j*	134	*k*	456	*l*	2018
m	1223	*n*	3702	*o*	3697	*p*	1079	*q*	51	*r*	3124
s	3420	*t*	4450	*u*	1333	*v*	555	*w*	952	*x*	85
y	855	*z*	64								

Tabela III.7. Broj pojavljivanja pojedinih karaktera u tekstu na engleskom jeziku iz zadatka 3.1

a	8.38%	*b*	1.67%	*c*	3.18%	*d*	4.09%	*e*	12.21%	*f*	2.10%
g	1.90%	*h*	4.47%	*i*	7.57%	*j*	0.27%	*k*	0.91%	*l*	4.04%
m	2.45%	*n*	7.41%	*o*	7.40%	*p*	2.16%	*q*	0.10%	*r*	6.25%
s	6.84%	*t*	8.91%	*u*	2.67%	*v*	1.11%	*w*	1.91%	*x*	0.17%
y	1.71%	*z*	0.13%								

Tabela III.8. Procenat pojedinih karaktera u tekstu na engleskom jeziku iz zadatka 3.1

a	8	*b*	2	*c*	3	*d*	4	*e*	12	*f*	2
g	2	*h*	5	*i*	8	*j*	0	*k*	1	*l*	4
m	3	*n*	8	*o*	7	*p*	2	*q*	0	*r*	6
s	7	*t*	9	*u*	3	*v*	1	*w*	2	*x*	0
y	2	*z*	0								

Tabela III.9. Broj pojedinih karaktera u tipičnoj sekvenci formiranoj po uslovima zadatka 3.1

puta, *m* dva puta, *n* sedam puta, *o* sedam puta, *p* dva puta, *r* šest puta, *s* šest puta, *t* osam puta, *u* dva puta, *w* jednom i *y* jednom. To je ukupno 88 karaktera. Možemo dodati po jedno pojavljivanje za još 12 karaktera i odlučili smo da to budu onih 12 kojima najmanje nedostaje do cijelog procenta, što je sublimirano u Tabeli III.9.

3.2. Na sekvenci od $n = 100$ bita sa vjerovatnoćom jedinice $p = 0.2$ provjeriti stavove teoreme o asimptotskoj ekviparticionoj osobini i formirati odgovarajući tipični i visokovjerovatni skup. Provjeriti mogućnost uštede kodiranjem visokovjerovatnog skupa manjim brojem bita u odnosu na ostatak.

Rješenje: Vjerovatnoća tipičnog događaja sa 20 jedinica i 80 nula je: $p^{20}q^{80} \approx 1.853 \cdot 10^{-22}$. Broj ovakvih kombinacija je:

$$\binom{100}{20} \approx 5.36 \cdot 10^{20}.$$

Ukupna vjerovatnoća tipičnih događaja je $5.36 \cdot 10^{20} \cdot 1.853 \cdot 10^{-22} \approx 0.0993$. Entropija je $H(X) = 0.7219$, dok je $2^{nH(X)} = 5.38 \cdot 10^{21}$, a $2^{-nH(X)} = 1.856 \cdot 10^{-22}$. Dakle, uočljivo je sljedeće:

$$p^{20}q^{80} \approx 2^{-nH(X)}.$$

Dalje uočavamo da je broj elemenata u ovako definisanom tipičnom skupu $\binom{100}{20} \approx 5.36 \cdot 10^{20}$, što je znatno manje od $2^{nH(X)}$ (za red veličine). Međutim, tipični i visokovjerovatni skup ne čine samo elementi kod kojih se očekivani broj puta pojavljuju simboli (ovdje np i nq jedinica i nula respektivno) već su to i sekvence kod kojih važi približnost. Tako, na primjer, broj sekvenci sa 19 jedinica je $1.32 \cdot 10^{20}$, dok je broj sekvenci sa 21 jedinicom $2.04 \cdot 10^{21}$. Totalna vjerovatnoća sekvenci sa 19 jedinica je oko 0.0978, dok je totalna vjerovatnoća sekvenci sa 21 jedinicom 0.0945. Dakle, $2.72 \cdot 10^{21}$ sekvenci uzimaju ukupnu vjerovatnoću od 0.2916, u odnosu na ostale sekvence koje uzimaju preostalu vjerovatnoću 0.7084. Sve sekvence kod kojih se pojavljuje $k \in [14,27]$ jedinica imaju kumulativnu vjerovatnoću 0.919. Broj ovih sekvenci je $2.98 \cdot 10^{24}$, što je oko 2 milionita dijela

ukupnog broja sekvenci. Za kodiranje ovog broja sekvenci potrebno je oko $\log_2(2.98 \cdot 10^{24})$, što iznosi nešto preko 81, odnosno 82 bita. Dodajmo još jedan (prefiksni) bit, pa sekvence visokovjerovatnog skupa kodirajmo sa 83 bita, a ostale sekvence kodirajmo sa $100+1$ bit (dodatni bit je prefiks). Prosječna dužina kodne riječi kojom kodiramo 100 bita tada iznosi

$$\bar{L}_{100} = 0.919 \times 83 + 0.081 \times 101 \approx 84.6 \text{ bita}.$$

Dakle, ostvarena ušteda na ovaj način je oko 15.4%, što nije zanemarljivo. Ipak, treba uočiti da entropija kaže da se može postići prosječna dužina koja je $\geq 100H(X) = 72.19$ bita, odnosno da imamo dosta prostora za dalje unapređenje, a drugi problem je u činjenici da nije jednostavno kodiranje dugačkih sekvenci (u ovom slučaju od 100 bita). Ovim primjerom je pokazano i da se povećanjem dužine kodne riječi približavamo dostižnim granicama, uz cijenu usložnjavanja kodnog postupka.

3.3. Za dokazivanje različitih stavova u teoriji informacija, uključujući one vezane za asimptotsku ekviparticionu osobinu, kao i one koje pokazuju mogućnost postizanja kompresije podataka, često se koristi Ćezarova teorema (ital. *Ernesto Cesaro* – poznati italijanski matematičar s prelaza iz XIX u XX vijek). Teorema glasi: Ako niz brojeva a_n konvergira po vjerovatnoći ka a, tada suma $b_k = \frac{1}{k}\sum_{i=1}^{k} a_i$ konvergira a. Dokazati predmetnu teoremu.

Dokaz. Ako je a_n blisko a, srednja vrijednost prvih n brojeva takođe teži a, zato što, kako se povećava broj elemenata, početni elementi imaju sve manji uticaj na srednju vrijednost. Pošto $a_n \to a$, to znači da za neko $\varepsilon > 0$ postoji $N(\varepsilon)$, tako da za $n \geq N(\varepsilon)$ važi $|a_n - a| \leq \varepsilon$. Dalje slijedi:

$$|b_n - a| = \left| \frac{1}{n}\sum_{i=1}^{n} a_i - a \right| \leq \frac{1}{n}\left| \sum_{i=1}^{n} a_i - a \right|.$$

Predmetna relacija slijedi na osnovu poznate nejednakosti trougla, odnosno da je suma distanci veća od svake pojedinačne distance:

$$\frac{1}{n}\left| \sum_{i=1}^{n} a_i - a \right| \leq \frac{1}{n}\left| \sum_{i=1}^{N_\varepsilon} a_i - a \right| + \frac{(n+N)}{n}\varepsilon \leq \frac{1}{n}\left| \sum_{i=1}^{N_\varepsilon} a_i - a \right| + \varepsilon.$$

Prvi član u ovom izrazu teži nuli ako n teži beskonačnosti. Naime, broj članova u sumi je ograničen, dok n može da raste. Dakle, iz ovog se može lako zaključiti da se razlika između b_n i a može učiniti proizvoljno malom.

3.4. Entropijski odnos za stohastički proces $\{X_i\}$ definiše se kao:

$$H(\chi) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, ..., X_n),$$

pod pretpostavkom da postoji granična vrijednost. Slična veličina se može usvojiti za uslovnu entropiju:

$$H'(\chi) = \lim_{n\to\infty} \frac{1}{n} H(X_n \mid X_{n-1}, X_{n-2}, ..., X_1).$$

Potrebno je pokazati da kod stacionarnih procesa ove dvije veličine konvergiraju, odnosno da su jednake:

$$H'(\chi) = H(\chi).$$

Rješenje: Po lančanom pravilu slijedi:

$$\frac{H(X_1, X_2, ..., X_n)}{n} = \frac{1}{n} H(X_n \mid X_1, X_2, ..., X_n).$$

Poznato je da entropije imaju granične vrijednosti. Po Ćezarovoj sredini (prethodna teorema) srednja vrijednost ima istu graničnu vrijednost, koja je u ovom slučaju jednaka granici uslovne entropije:

$$H(\chi) = \lim_{n\to\infty} \frac{H(X_1, X_2, ..., X_n)}{n} = \lim_{n\to\infty} H(X_n \mid X_1, X_2, ..., X_n) = H'(\chi).$$

3.5. Pretpostaviti da je poznata entropija izvora $H(X)$. Neka je nad alfabetom izvora primijenjena linearna transformacija i neka je na osnovu kodnog simbola x dobijen kodni simbol $y = ax + b$ gdje su a i b konstante. Odrediti entropiju izvora y u zavisnosti od entropije izvora x.

Rješenje: Entropija transformisanog događa je

$$H(Y) = -\sum_{y} p(y) \log p(y).$$

Međutim, kako je linearna transformacija preslikavanje 1 na 1, jednoj vrijednosti x odgovara jedna vrijednost y, jasno da je $H(Y) = H(X)$.

3.6. Razmotriti slučaj slanja sekvence od $n = 10$ i.i.d. simbola ako se simbol 0 pojavljuje s vjerovatnoćama $p = 0.1$, 0.2 i 0.4, respektivno u smislu asimptotske ekviparticione osobine.

Rješenje: Tabele III.10–III.12 prikazuju ekvivalent Tabele III.1 za $p = 0.1$, $p = 0.2$ i $p = 0.4$, respektivno. Iz Tabele III.10 vidimo da je tipična sekvenca ona sa 9 jedinica i da se pojavljuje s ukupnom vjerovatnoćom 0.3874. Entropija ovog događaja je $H(X) = 0.459$, $nH(X) = 4.59$, pa je $2^{-nH(X)} = 0.03874$, što je praktično isto kao ono što piše u tabeli. Za dodatnu provjeru posmatrajmo da li je moguće ostvariti neku

r	$0.1^r 0.9^{10-r}$	$10!/[r!(10-r)!]$	**Ukupno**
0	1×10^{-10}	1	1×10^{-10}
1	9×10^{-10}	10	9×10^{-9}
2	8.1×10^{-9}	45	3.645×10^{-7}
3	7.29×10^{-8}	120	8.75×10^{-6}
4	6.561×10^{-7}	210	1.378×10^{-4}
5	5.9×10^{-6}	252	0.00149
6	5.314×10^{-5}	210	0.01116
7	4.783×10^{-4}	120	0.0574
8	0.0043	45	0.1937
9	**0.0387**	**10**	**0.3874**
10	0.3487	1	0.3487

Tabela III.10. Broj jedinica u sekvenci, vjerovatnoća pojedinačne sekvence i ukupna vjerovatnoća poruka sa r jedinica za $p = 0.1$

R	$0.2^r 0.8^{10-r}$	$10!/[r!(10-r)!]$	**Ukupno**
0	1.024×10^{-7}	1	1.024×10^{-7}
1	4.096×10^{-7}	10	4.096×10^{-6}
2	1.6384×10^{-6}	45	7.373×10^{-5}
3	6.554×10^{-6}	120	7.864×10^{-4}
4	2.6214×10^{-5}	210	0.0055
5	1.0486×10^{-4}	252	0.0264
6	4.1943×10^{-4}	210	0.0881
7	0.0017	120	0.2013
8	**0.0067**	**45**	**0.3020**
9	0.0268	10	0.2684
10	0.1084	1	0.1084

Tabela III.11. Broj jedinica u sekvenci, vjerovatnoća pojedinačne sekvence i ukupna vjerovatnoća poruka sa r jedinica za $p = 0.2$

kompresiju korišćenjem najprimitivnijih oblika kodiranja na ovom kodu. U tipičnom skupu sekvenci je 10. Visokovjerovatni skup dobijamo proširivanjem tipičnog skupa najvjerovatnijom sekvencom sa svim jedinicama, koja ima vjerovatnoću 0.3487, i putem pet sekvenci iz skupa sa osam jedinica, s ukupnom vjerovatnoćom $5 \times 0.0043 = 0.0215$. Ukupna vjerovatnoća ovih 16 događaja je $0.3874 + 0.3487 + 0.0215 = 0.7576$. Ovih 16 simbola se mogu kodirati sa 4 bita (plus jedan bit prefiksa), dok se preostalih 1008 simbola mogu kodirati sa 10 bita (plus jedan bit prefiksa), tako da je prosječna dužina kodne riječi:

$$0.7576 \times 5 + 0.2424 \times 11 = 6.4544 \text{ bita/sekvenci.}$$

r	$0.4^r 0.6^{10-r}$	$10!/[r!(10-r)!]$	**Ukupno**
0	1.0486×10^{-4}	1	1.0486×10^{-4}
1	1.5927×10^{-4}	10	0.001573
2	2.359×10^{-4}	45	0.010617
3	3.5389×10^{-4}	120	0.042467
4	5.3084×10^{-4}	210	0.111476
5	7.9626×10^{-4}	252	0.200658
6	**0.00119439**	**210**	**0.250823**
7	0.0017916	120	0.214991
8	0.0026874	45	0.120932
9	0.0040311	10	0.040311
10	0.0060466	1	0.0060466

Tabela III.12. Broj jedinica u sekvenci, vjerovatnoća pojedinačne sekvence i ukupna vjerovatnoća poruka sa r jedinica za $p = 0.4$

Dakle, u stanju smo da ovim jednostavnim postupkom prištedimo 35.45% u kodiranju za $p = 0.1$ korišćenjem čak i ovako primitivne kodne šeme. Ponovimo proceduru za $p = 0.2$. Imamo da je sekvenca sa dvije nule i osam jedinica tipična. Entropija ovog skupa je $H(X) = 0.7219$, dok je $nH(X) = 7.219$. Tada je $2^{-nH(X)} = 0.0067$ i potpuno tačno odgovara vjerovatnoći pojavljivanja pojedinačne poruke iz tipičnog skupa! Analizirajmo mogućnost kompresije. Poruka u tipičnom skupu je 45, a do najbližeg stepena dvojke je 19 ($2^6 = 64$). Dodajmo svih 10 sekvenci sa jednom nulom i jednu sekvencu sa svim jedinicama, kao i 8 sekvenci sa 3 nule. Ukupna vjerovatnoća ovog visokovjerovatnog skupa je sada:

$$0.3020 + 0.2684 + 0.1074 + 8 \times 0.0017 = 0.6912.$$

Za kodiranje ove sekvence potrebno je 6 bita plus jedan bit kao prefiks, dok je za ostale simbole, s ukupnom vjerovatnoćom 0.3088, potrebno 10 bita i jedan kao prefiks:

$$0.6912 \times 7 + 0.3088 \times 11 = 8.2354 \text{ bita.}$$

Ušteda je sada manja nego u slučaju kada je tipična sekvenca imala manji broj članova i kada je bila karakterističnija po svom obliku. Vidjećemo da u narednom slučaju, kada p raste i kada se primiče $p = 0.5$, dolazi do smanjivanja uštede koja se dobija putem asimptotske ekviparticione osobine, odnosno u opštem slučaju putem kodiranja. Dakle, kada su simboli s uniformnijim raspodjelama, dobijamo manju dobit.

Posmatrajmo sada Tabelu III.12 za $p = 0.4$. Imamo da je tipična sekvenca sa 4 nule i 6 jedinica, koja se pojavljuje s ukupnom vjerovatnoćom 0.250823. Entropija

ovog skupa je $H(X) = H(0.4) = 0.971$, a dalje imamo $nH(X) = 9.71$, pa je $2^{-nH(X)} = 0.00119$, a ovo se tek na osmu cifru razlikuje od vjerovatnoće pojedinačnog člana tipičnog skupa. Dopunimo ovaj skup sa još četiri najvjerovatnija elementa (sve jedinice i 3 sekvence sa 9 jedinica) da bismo skupili $256 = 2^8$ sekvenci. Vjerovatnoća ove visokovjerovatne sekvence je:

$$0.250823 + 0.0060466 + 3 \times 0.0040311 = 0.2689629.$$

Ako kodiramo sekvence visokovjerovatnog skupa sa $8 + 1$ bit (1 bit za predznak), a ostale sa $10 + 1$ bit, dobijamo da je prosječna dužina kodne riječi 10.462 bita po sekvenci od 10 simbola, čime ne ostvarujemo nikakav dobitak, već zapravo gubitak. Premda smo sofisticiranijim tehnikama kodiranja u stanju da ostvarimo dobitak, ovo nam pokazuje da kod alfabeta s ujednačenijim vjerovatnoćama pojedinačnih ili kombinacija simbola ne možemo da ostvarimo veliki dobitak kod kodiranja. Predmetni zaključak bi bio još izraženiji za slučaj $p = 0.5$, što se ostavlja za vježbu.

3.7. Posmatran je dokument na našem jeziku koji sadrži sljedeću raspodjelu simbola:

a	4694	*b*	589	*v*	1320	*g*	704	*d*	1463
đ	94	*e*	3409	ž	179	*z*	671	*i*	3734
j	1433	*k*	1505	*l*	1151	*lj*	166	*m*	1179
n	2324	*nj*	249	*o*	3517	*p*	1118	*r*	2143
s	1806	*t*	1729	ć	354	*u*	1582	*f*	159
h	210	*c*	439	č	390	*dž*	15	*š*	383.

Odrediti tipičnu sekvencu u slučaju da je $n = 100$. Neka je kodiranje datog fajla obavljeno na sljedeći način. Izdvojeno je K simbola koji se najčešće pojavljuju i kodirani su sa po $\log_2 K$ bita plus 0 kao prefiks, dok je ostatak kodiran po ASCII kodu i 1 kao prefiks. Odrediti koliko bita je potrebno za kodiranje čitavog fajla ako je $K = 2, 4, 8, 16$.

Rješenje: Prvi dio problema riješili smo tako što smo sumirali koliko ima karaktera u materijalu (38709), a zatim smo odredili procenat pojavljivanja svakog slova (prije glasa) i zaokružili na manji cjelobrojni procenat. Dobili smo 85 karaktera u tipičnom skupu, pa smo zatim 15 glasova, koji su najbliži većem procentu, uvećali za 1, čime smo dobili tipični skup u obliku:

a	12	*b*	1	*v*	3	*g*	2	*d*	4
đ	0	*e*	9	ž	0	*z*	2	*i*	10
j	4	*k*	4	*l*	3	*lj*	0	*m*	3
n	6	*nj*	1	*o*	9	*p*	3	*r*	6
s	5	*t*	4	ć	1	*u*	4	*f*	0
h	1	*c*	1	č	1	*dž*	0	*š*	1.

Dva najčešća simbola se pojavljuju 8408 puta. Ako izdvojimo ova dva simbola, prosječna dužina kodiranja je:

$$(8408 \times 2 + (38709 - 8408) \times 5)/38709 \approx 4.35 \text{ bita/simbolu.}$$

Četiri najčešća glasa pojavljuju se 15354 puta, pa je prosječna dužina kodiranja, ako njih izdvojimo:

$$(15354 \times 3 + (38709 - 15354) \times 5)/38709 \approx 4.21 \text{ bita/simbolu.}$$

Ponovimo postupak za slučaj izdvajanja 8 najčešćih simbola (pojavljuju se 23356 puta):

$$(23356 \times 4 + (38709 - 23356) \times 5)/38709 \approx 4.40 \text{ bita/simbolu.}$$

Za slučaj 16 najvjerovatnijih simbola proračun, zapravo, ne treba obavljati jer se podgrupa mora kodirati sa 5 bita, kao i ostatak u kome se nalazi 14 simbola. Dakle, u tom slučaju se ne ostvaruje nikakav dobitak, a vidimo da se on može ostvariti i ovako primitivnom kodnom šemom, a posebno veliki može biti pravilnim odabirom veličine grupe najvjerovatnijih simbola. Za vježbu testirati Hafmenov kod nad ovim alfabetom, te odrediti prosječnu dužinu kodne riječi i uporediti je s procijenjenom entropijom ovog sistema.

3.8. Posmatrajte model opisan zadatkom 3.2. Pretpostaviti da je K najčešće pojavljivanih simbola kodirano sa $\log_2 K$ bita i prefiksom 1, dok je ostatak simbola kodiran sa ASCII kodom sa prefiksnim bitom 0. Koliko je bita potrebno za zapis originalne sekvence u ASCII kodu, kao i za zapis komprimovane sekvence, ako je $K = 2, 4, 8, 16$.

Rješenje: Dva najčešće pojavljivana simbola su *e*, s vjerovatnoćom 12.21%, i *t*, sa vjerovatnoćom 8.91%. Ako njih kodiramo sa 1 bitom i dodatnim jednim prefiksnim bitom 1, a ostala 24 simbola kodiramo sa 5 bita i jednim prefiksnim, dobijamo prosječnu dužinu kodne riječi:

$$2 \times 0.2112 + 6 \times 0.7888 = 5.1552 \text{ bita/simbolu.}$$

Naredna dva najvjerovatnija simbola su: *a*, sa 8.38%, i *i*, sa vjerovatnoćom 7.57%. Dakle, četiri najvjerovatnija simbola u tom slučaju kodiramo sa 2 bita i jednim za prefiks, dok ostala 22 moramo kodirati sa 5 plus 1 za prefiks. Prosječna dužina kodne riječi u ovom slučaju je:

$$3 \times 0.3807 + 6 \times 0.6193 = 4.8579 \text{ bita/simbolu.}$$

Naredna 4 najvjerovatnija simbola imaju vjerovatnoće: 7.41%, 7.40%, 6.84% i 6.25%. Tada je ukupna vjerovatnoća 8 najvjerovatnijih simbola: 56.97% i moraju

se kodirati sa 3 + 1 bit, dok preostalih 18 imaju vjerovatnoću 43.03% i kodiraju se sa 5 + 1 bit. Prosječna dužina kodne riječi je:

$$4 \times 0.5697 + 6 \times 0.4303 = 4.8606 \text{ bita/simbolu.}$$

Konačno, situacija sa pridavanjem narednih 8 simbola je trivijalna, jer 16 simbola u glavnoj grupi moramo kodirati sa 4 + 1 bit baš kao i preostale simbole u grupi sa malim vjerovatnoćama. Kao što vidimo, najbolji rezultat se postiže ako je odabrano kodiranje 4 najvjerovatnija simbola odvojeno od kodiranja preostala 22 simbola. Vidimo da je ušteda simbolična, ali postoji čak i kod ovog krajnje primitivnog načina kodiranja, koji je upotrijebljen samo da ilustruje AEP.

3.9. Pretpostavimo da nisu poznate tačne vrijednosti vjerovatnoća pojavljivanja pojedinih kodnih simbola, već da se prilikom njihove procjene pravi greška e_i. Odrediti razliku u prosječnoj dužini kodne riječi u ovom slučaju, kao i varijansu greške koja se pravi.

Rješenje: Neka su p_i tačne vrijednosti vjerovatnoća u procesu dizajna Hafmenovog koda. Neka su:

$$p_i' = p_i + e_i$$

vjerovatnoće koje su korišćene u dizajnu koda (e_i je greška u procjeni pojavljivanja pojedinih simbola). Očigledno važi: $\frac{1}{q}\sum_{i=1}^{q} e_i = 0$. Neka je:

$$\frac{1}{q}\sum_{i=1}^{q} e_i^2 = \sigma^2$$

mjera greške u procjeni vjerovatnoće kodnih simbola. Odredimo prosječnu dužinu koda:

$$L' = \frac{1}{q}\sum l_i p_i' = \frac{1}{q}\sum l_i p_i + \frac{1}{q}\sum l_i e_i = L + \frac{1}{q}\sum l_i e_i .$$

Posljednji član je promjena u dužini prosječne kodne riječi. Da bismo odredili ekstremne vrijednosti ovog izraza, koristimo metod Lagranžovih množilaca i prethodna dva ograničenja:

$$\Lambda = \frac{1}{q}\sum l_i e_i - \lambda\frac{1}{q}\sum e_i - \mu\left(\frac{1}{q}\sum e_i^2 - \sigma^2\right).$$

Odredimo parcijalne izvode po e_i i izjednačimo sa nulom:

$$\frac{\partial\Lambda}{\partial \mathrm{e_i}} = \frac{1}{q}(l_i - \lambda - 2\mu e_i) = 0.$$

Kada se ove jednačine sumiraju za svako i, dobija se: $\lambda = \frac{1}{q}\sum_i l_i$. Ako se jednačine pomnože sa e_i i sumiraju, dobija se:

$$\frac{1}{q}\sum_i l_i e_i - 2\mu\frac{1}{q}\sum_i e_i^2 = 0.$$

Na osnovu ove jednakosti dobijamo izraz za μ. Množeći prethodnu jednačinu sa l_i i sumirajući po i, uz uvrštavanje vrijednosti za λ i μ iz prethodnih jednačina, dobijamo:

$$\frac{1}{q}\sum_i l_i^2 - \left(\frac{1}{q}\sum_i l_i\right)^2 - \frac{\left(\frac{1}{q}\sum_i e_i l_i\right)^2}{\frac{1}{q}\sum_i e_i^2} = 0$$

$$\left(\frac{1}{q}\sum_i e_i l_i\right)^2 = \left[\frac{1}{q}\sum_i l_i^2 - \left(\frac{1}{q}\sum_i l_i\right)^2\right]\sigma^2 = \mathrm{var}[l_i]\,\mathrm{var}[e_i].$$

Na osnovu ovoga možemo zaključiti da mala greška u procjeni vjerovatnoće pojavljivanja simbola neće značajno promijeniti prosječnu dužinu kodne riječi u odnosu na optimalnu sve dok varijansa l_i nije ekstremno velika. Ovo ukazuje na potrebu da, u slučaju da u procesu Hafmenovog kodiranja imamo mogućnost da biramo simbole sa više istih vjerovatnoća, selektujemo one koji su se u dosadašnjem postupku kodirali sa manjim brojem bita, kako bismo smanjili najduže kodne riječi, odnosno smanjili varijansu l_i.

3.10. Dati su kodovi (a)−(e). Da li su ovo jednoznačno dekodabilni kodovi i da li su u pitanju trenutni kodovi?

Kodovi	(a)	(b)	(c)	(d)	(e)
s_1	00	0	0	0	0
s_2	01	100	10	100	10
s_3	10	110	110	110	110
s_4	11	111	111	11	11

Za date kodove provjeriti važenje Kraftove nejednakosti.

Rješenje: Kraftova nejednakost za jednoznačno dekodabilne kodove glasi:

$$\sum_{i=1}^{m} D^{-l_i} \le 1,$$

gdje je D broj simbola izlaznog alfabeta, dok je l_i dužina i-te kodne riječi. Svih pet kodova su binarni, odnosno $D = 2$. Kod prvog koda (a) sve kodne riječima imaju dužinu 2, pa važi:

$$4 \cdot \frac{1}{2^2} = 1 \leq 1,$$

što je potreban uslov da je kod jednoznačno dekodabilan (što i jeste, jer predstavlja standardno binarno kodiranje četiri simbola).

Kod drugog koda (b) važi:

$$\frac{1}{2} + 3 \cdot \frac{1}{2^3} = \frac{7}{8} \leq 1,$$

što ponovo znači da bi u pitanju mogao biti jednoznačno dekodabilan kod, što i jeste.

Kod trećeg koda (c) važi:

$$\frac{1}{2} + \frac{1}{4} + 2 \cdot \frac{1}{2^3} = 1 \leq 1.$$

Odnosno, zadovoljen je potreban uslov, a kod je zaista jednoznačno dekodabilan jer je u pitanju koma kôd kod kojeg nula ili treći bit označava kraj koda za neki simbol.

Za kod (d) važi:

$$\frac{1}{2} + \frac{1}{4} + 2 \cdot \frac{1}{2^3} = 1 \leq 1.$$

Dakle, ispunjen je potreban uslov za jednoznačnu dekodabilnost. I zaista kod je jednoznačno dekodabilan, ali nije trenutni kod, odnosno ne može se dekodirati simbol nakon prijema posljednjeg bita kojim je kodiran. Pokušajte da razvijete dekodirajuće pravilo za ovaj kod.

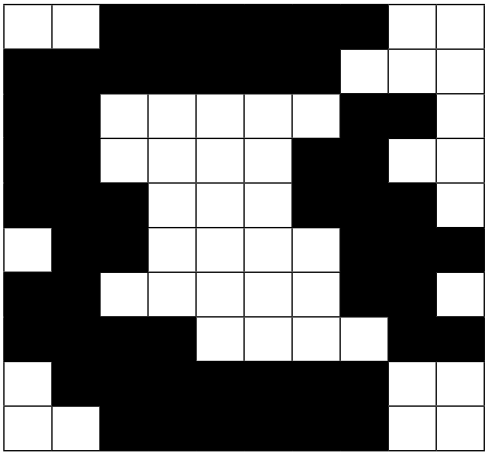
Kod (e) nije jednoznačno dekodabilan jer nije zadovoljena Kraftova nejednakost:

$$\frac{1}{2} + 2 \cdot \frac{1}{4} + \frac{1}{2^3} = \frac{9}{8} > 1.$$

3.11. Razmotriti jednoznačnu dekodabilnost i optimalnost kodova datih u Tabeli III.13 ako se pojedini simboli alfabeta pojavlju sa vjerovatnoćama $P(s_1) = 1/2$, $P(s_2) = 1/4$, $P(s_3) = 1/8$, $P(s_4) = 1/8$.

S	Kod (a)	Kod (b)	Kod (c)
s_1	00	0	0
s_2	01	10	01
s_3	10	110	011
s_4	11	1110	0111

Tabela III.13. Kodovi definisani nad alfabetom S



Slika III.12. Primjer faks dokumenta iz zadatka 3.12

Rješenje: Svi kodovi su jednoznačno dekodabilni, ali nijedan nije optimalan. Za kodove (b) i (c) ovo je jasno iz Leme III.1, jer dvije najduže riječi u jednoznačno dekodabilnom kodu moraju biti iste dužine, što ovdje nije slučaj. Kod (b) je trenutni, što se ne može reći za kod (c) koji mora da sačeka da se primi još neki bit da bismo bili sigurni u primljenu poruku. Na primjer, nakon 01 mora se sačekati da li će se primiti 0 pa da znamo da je prethodno bilo s_2 ili dobijamo 1, nakon čega i dalje imamo neodređenost. Ove opservacije ne zavise od vjerovatnoće pojavljivanja simbola. Kod (a) je standardni kod sa 2 bita/simbolu, ali npr. kod (b), skraćujući posljednju kodnu riječ (za s_4) na tri bita, daje u predmetnom slučaju kompaktan kod koji ima prosječnu dužinu kodne riječi:

$$\bar{L} = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{4} = 1.75 \text{ bita/simbolu}.$$

3.12. Data je binarna slika (Slika III.12) koja može da predstavlja djelić faks dokumenta. Razmotriti kako se predmetna slika može kodirati postupkom koji je sličan RLE kodu.

Rješenje: Originalni kod zahtijeva 100 bita za kodiranje ove površi, dimenzija 10×10. Pošto ovdje nema drugih simbola osim jedinica i nula, mi ćemo kodiranje obaviti tako što ćemo kodirati prvi simbol i zatim samo pisati koliko puta se taj simbol pojavi zaredom (nastavljajući na naredni red):

1 2 6 2 7 3 2 5 2 1 2 4 2 2 3 3 3 2 2 4 5 5 2 1 4 4 2 1 7 4 6 2.

Kao što vidimo, imamo 32 simbola umjesto 100 bita i na prvi pogled smo napravili značajnu uštedu. Međutim, sada umjesto bita imamo nebinarne simbole. Pretpostavimo da prvi simbol kodiramo sa 1 bitom, a ostale da kodiramo sa 3, dobijamo da je potrebno za predmetni kod $1 + 31 \times 3 = 94$ bita, a to znači da je ušteda samo

6% uz dodatno procesiranje. Stoga, kodiranje faks dokumenata, na primjer, podrazumjeva i gledanje korelacije prethodnih redova. Na primjer, redovi od trećeg do osmog se malo razlikuju, kao što se međusobno malo razlikuju deveti i deseti red i prvi i drugi. Pored toga, na riječ koja je data gore možemo provesti Hafmenovo kodiranje. Posebno dizajniran kod koji se bavi ovom problematikom naziva se READ (engl. *Relative Element Address Designate*).

3.13. Koji dekadni broj je predstavljen trobitnim Grejovim kodom 110101? Koji broj je prikazan ovim nizom bita ako je u pitanju šestobitni Grejov kod?

Rješenje: U prvom slučaju imamo dva simbola 110 i 101. Prvi bit u originalnoj poruci je isti kao u kodiranoj, a naredni su jednaki $a_i = b_i \oplus a_{i-1}$, gdje je b_i tekući bit Grejovog koda, dok je a_{i-1} prethodni dekodirani bit. Primjenjujući navedenu operaciju, dobijamo za uvedene trobitne simbole:

100_2 110_2, odnosno 46 dekadno zapisano.

Pretpostavljajući da je 110101 šestobitni Grejov kod, primjenjujući istu proceduru u dekodiranju, dobijamo:

$$100110_2 = 38_{10}.$$

3.14. Kodirati sekvence RLE kodom i odrediti faktor kompresije u oba slučaja:

a)	15	15	15	17	17	17	17	17	21	21	21
	21	21	21	21	6	23	23	34	34	34	34
	11	11	11	11							
b)	15	16	15	17	17	17	16	17	21	21	21
	22	21	21	21	6	23	23	34	33	34	34
	11	12	11	11.							

Rješenje: Sekvenca a) se može kodirati RLE kodom kao:

(15,3) (17,5) (21,7) (6,1) (23,2) (34,4) (11,4).

Polazna sekvenca ima 26 simbola, dok kodirana ima 14 simbola (sedam parova) čime je ostvarena kompresija na 14/26 = 53.8% originalne informacije.

Kodirajmo sada sekvencu b):

(15,1) (16,1) (15,2) (17,3) (16,1) (17,1) (21,3) (22,1) (21,3)
(6,1) (23,2) (34,1) (33,1) (34,2) (11,1) (12,1) (11,2).

Dobili smo sekvencu sa 34 simbola, odnosno ostvarili smo tzv. negativnu kompresiju. Ovo ukazuje na činjenicu da se RLE kod koristi samo kod specifičnih signala ili u kombinaciji s drugim kodovima.

3.15. Simboli s_1 i s_2 u alfabetu se pojavljuju sa vjerovatnoćama 7/8 i 1/8. U sekvenci, ovi simboli se pojavljuju kao i.i.d. procesi. Odrediti prosječnu dužinu kodne riječi kada se izvrši kodiranje pojedinačnih simbola, kombinacije dva simbola i kombinacije tri simbola. Izvršiti poređenje s entropijom.

Rješenje: Entropija ovog sistema je: $H(X)=H(1/8)\approx 0.5436$. Ako posmatramo pojedinačne simbole, očigledno ostvarujemo kodiranje sa 1 bit/simbolu. Kada vršimo kodiranje sa parovima simbola, imamo četiri moguće kombinacije. Ovdje dajemo Hafmenovo kodiranje bez navođenja pojedinačnih koraka, pošto su, po našem mišljenju, očigledni:

S^2	P_i	X_i
s_1s_1	49/64	0
s_1s_2	7/64	10
s_2s_1	7/64	110
s_2s_2	1/64	111.

Entropija ovog skupa je:

$$H(S^2)=2H(X)=-\frac{49}{64}\log_2\frac{49}{64}-\frac{7}{64}\log_2\frac{7}{64}-\frac{7}{64}\log_2\frac{7}{64}-\frac{1}{64}\log_2\frac{1}{64}\approx 1.0871.$$

Prosječna dužina kodne riječi za kodiranje dva simbola je:

$$l_2=1\cdot\frac{49}{64}+2\cdot\frac{7}{64}+3\cdot\left(\frac{7}{64}+\frac{1}{64}\right)=\frac{87}{64}\approx 1.3594.$$

Dakle, prosječna dužina za jedan simbol je $l=l_2/2\approx 0.6797$, čime smo se znatno približili uslovima kodne teoreme. Ako idemo sada korak dalje i kodiramo tri simbola, imamo sljedeću situaciju:

S^3	P_i	X_i
$s_1s_1s_1$	343/512	1
$s_1s_1s_2$	49/512	011
$s_1s_2s_1$	49/512	010
$s_2s_1s_1$	49/512	001
$s_1s_2s_2$	7/512	00011
$s_2s_1s_2$	7/512	00010
$s_2s_2s_1$	7/512	00001
$s_2s_2s_2$	1/512	00000.

Entropija i.i.d. sekvence od tri simbola je:

$$H(S^2)=3H(X)\approx 1.6307,$$

dok je prosječna dužina kodne riječi koju smo postigli Hafmenovim postupkom:

$$l_3 = 1\cdot\frac{343}{512} + 3\cdot\left(\frac{49}{512}+\frac{49}{512}+\frac{49}{512}\right) + 5\cdot\left(\frac{7}{512}+\frac{7}{512}+\frac{7}{512}+\frac{1}{512}\right) = \frac{894}{512} \approx 1.746.$$

Odnosno, po jednom simbolu imamo:

$$l = l_3/3 \approx 0.582.$$

Dakle, prosječna dužina kodne riječi se značajno primakla entropiji sistema, ali uz cijenu što naš alfabet sada ima više simbola, prouzrokujući time veće memorijske i računske zahtjeve.

3.16. Temperatura je mjerena u cijelim stepenima na svaki sat jednog dana:

6 5 4 2 0 – 2 – 1 – 1 0 1 2 3 5 7 9 11 12 10 9 8 7 7 6 5.

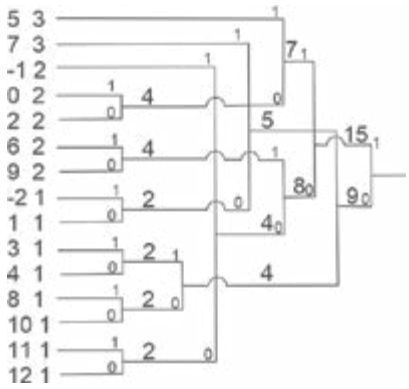
Dobijene vrijednosti treba kodirati diferencijalnim kodom, pa na taj kod primijeniti Hafmenov postupak i porediti dobijenu prosječnu dužinu kodne riječi sa situacijom da kodiranje nije vršeno, odnosno da je vršeno samo putem Hafmenovog koda bez primjene diferencijalnog kodiranja.

Rješenje: Prilikom rješavanja zanemarićemo praktične probleme koji se ovdje mogu pojaviti vezano za predznanje dekodera o postupku kodiranja. Vidimo da imamo 24 simbola koji se nalaze u domenu [– 2,12], odnosno domenu koji ima 15 mogućih vrijednosti. Dakle, bez primjene bilo kakvog postupka kompresije, možemo da kodiramo ovaj skup sa:

$$24 \times 4 = 96 \text{ bita.}$$

Vrijednost	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
Učestanost	1	2	2	1	2	1	1	3	2	3	1	2	1	1	1

Tabela III.14. Učestanost pojavljivanja vrijednosti temperatura



Slika III.13. Hafmenovo kodiranje temperatura

Učestanost pojavljivanja simbola alfabeta je data u Tabeli III.14. Primijenimo Hafmenov kodni postupak sa kodnim stablom prikazanim na Slici III.13. Vidimo da su prva tri kodna simbola (5, 7 i -1), koji se pojavljuju 8 puta, ukupno kodirani sa 3 bita, posljednja dva (11 i 12) sa pet bita, a ostali su kodirani sa 4 bita. Dakle, ukupan broj bita koji je upotrijebljen za kodiranje Hafmenovom kodnom šemom je:

$$8\times 3+2\times 5+14\times 4=88 \text{ bita.}$$

Ostvarena je ušteda od samo 8 bita, uz izazov da moramo da obezbijedimo da prijemnik zna postupak kodiranja, što može da bude memorijski skuplje od uštede (međutim, ovu problematiku ostavimo po strani).

Alternativna varijanta je primjena diferencijalnog kodiranja na polaznu sekvencu:

$$6\ -1\ -1\ -2\ -2\ -2\ 1\ 0\ 1\ 1\ 1\ 1\ 2\ 2\ 2\ 2\ 1\ -2\ -1\ -1\ -1\ 0\ -1\ -1.$$

U Tabeli III.15 prikazane su učestanosti vrijednosti koje se pojavljuju u ovoj poruci koja je kodirana diferencijalnim kodom. Potreban broj bita za zapis ovakve poruke (ponovo bez ulaženja u problematiku prenosa koda) je (Slika III.14):

$$2\times 17+3\times 4+4\times 3=58\text{bita},$$

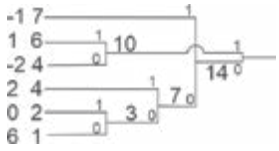
što je već značajna ušteda koja se ostvaruje kod sporopromjenljivih sekvenci.

3.17. Alfabet ima 9 kodnih simbola sa vjerovatnoćama: 0.23, 0.17, 0.16, 0.14, 0.12, 0.10, 0.03, 0.03, 0.02. Izvršiti kodiranje ternarnim (sa tri simbola) Hafmenovim kodom. Izvršiti kodiranje Hafmenovim kodom sa četiri simbola. Odrediti za svaki simbol odgovarajući kod, kao i prosječne dužine kodnih riječi u oba slučaja.

Rješenje: Neka su $\{a, b, c\}$ simboli ternarnog Hafmenovog koda. Potreban je odrediti broj kodnih simbola najmanje vjerovatnoće koji se kodiraju u prvom koraku. To se određuje na sljedeći način: od ukupnog broja kodnih poruka oduzima se po D–1 u svakom koraku ($D=3$ je broj simbola u alfabetu Hafmenovog koda). Broj simbola koje kodiramo u početnom koraku određujemo sljedećom procedurom:

Vrijednost	-2	-1	0	1	2	6
Učestanost	4	7	2	6	4	1

Tabela III.15. Učestanost pojavljivanja vrijednosti u sekvenci temperatura kodiranoj diferencijalnim kodom



Slika III.14. Hafmenov kod primijenjen nad diferencijalnim kodom

$$D=3 \text{ i } Q=9$$

$$Q-(D-1)=Q \qquad 9-2=7 \quad 7-2=5 \quad 5-2=3$$

$3 \le 3 \;\Rightarrow$ u prvom koraku kodiraju se 3 simbola.

Na Slici III.15 prikazan je postupak kodiranja. Simboli su kodirani na sljedeći način:

s_1: *a*	s_4: *bc*	s_7: *cca*
s_2: *ba*	s_5: *ca*	s_8: *ccb*
s_3: *bb*	s_6: *cb*	s_9: *ccc*.

Prosječna dužina kodne riječi je:

$$\bar{L}=\sum_{i=1}^{N} l_i p_i = 0.23+2(0.17+0.16+0.14+0.12+0.1)+3(0.03+0.03+0.02)=$$
$$=0.23+1.38+0.24=1.85$$

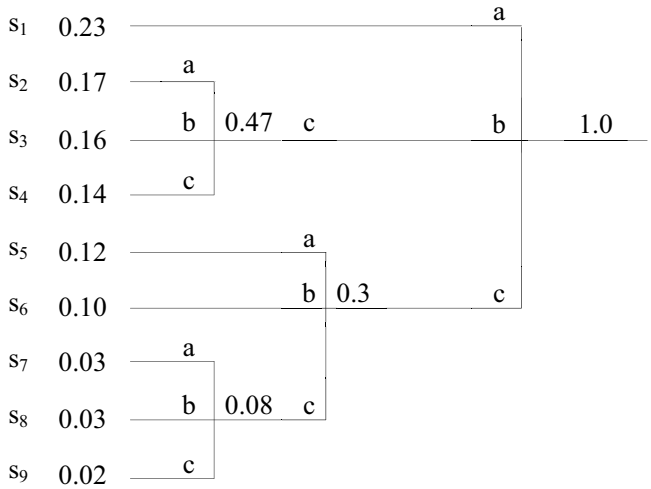
izraženo u simbolima izlaznog (ternarnog) po simbolu ulaznog (originalnog) alfabeta.

U slučaju kodiranja kvaternarnim alfabetom, u prvom koraku obuhvatamo:

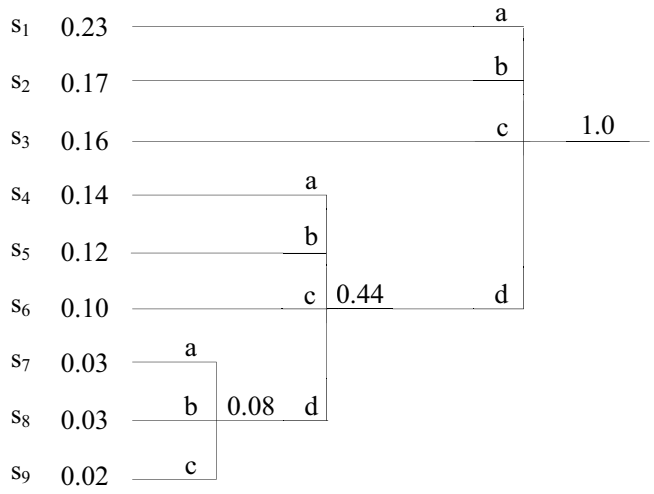
$$D=4 \text{ i } Q=9$$

$$Q-(D-1)=Q \qquad 9-3=6 \quad 6-3=3$$

$3 \le 4 \;\Rightarrow$ u prvom koraku kodiramo 3 simbola ulaznog alfabeta.



Slika III.15. Hafmenovo kodiranje ternarnim alfabetom



Slika III.16. Hafmenovo kodiranje kvaternarnim alfabetom

Na Slici III.16 prikazan je postupak kodiranja Hafmenovim kvaternarnim kodom, koji rezultuje u sljedećoj kodnoj šemi (izlazni alfabet je $\{a, b, c, d\}$):

s_1: *a* s_4: *da* s_7: *dda*
s_2: *b* s_5: *db* s_8: *ddb*
s_3: *c* s_6: *dc* s_9: *ddc*.

Prosječna dužina kodne riječi za kvaternarni Hafmenov kod je:

$$\bar{L} = \sum_{i=1}^{N} l_i p_i = 0.23 + 0.17 + 0.16 + 2(0.14 + 0.12 + 0.1) + 3(0.03 + 0.03 + 0.02)$$
$$= 0.56 + 0.72 + 0.24 = 1.52,$$

mjereno u simbolima kvaternarnog alfabeta po simbolima ulaznog alfabeta. Entropija ovog sistema je:

$$H(X) = -\sum_{i=1}^{9} p_i \log_2 p_i \approx 2.858 \text{bita / simbolu.}$$

Ne treba misliti da smo probili uslove prve Šenonove teoreme jer smo entropiju mjerili različitom mjerom u odnosu na prosječnu dužinu kodne riječi. Da bismo entropiju sveli na istu mjeru, treba podijeliti ovu relaciju sa $\log_2 4 = 2$, čime dobijamo $H(X)/2 \approx 1.429$.

3.18. U Tabeli III.16 data je vjerovatnoća simbola engleskog alfabeta iz nekog teksta. Izvršiti Hafmenovo kodiranje ovog alfabeta.

Rješenje: Slika III.17 prikazuje formiranje Hafmenovog koda (stabla) za predmetni alfabet. Dobijene kodne riječi su sublimirane u Tabeli III.17. Prosječna dužina kodne riječi koju dobijamo ovim postupkom je:

$$H(X) = -\sum_{i=1}^{9} p_i \log_2 p_i \approx 2.858 \text{ bita/simbolu}$$

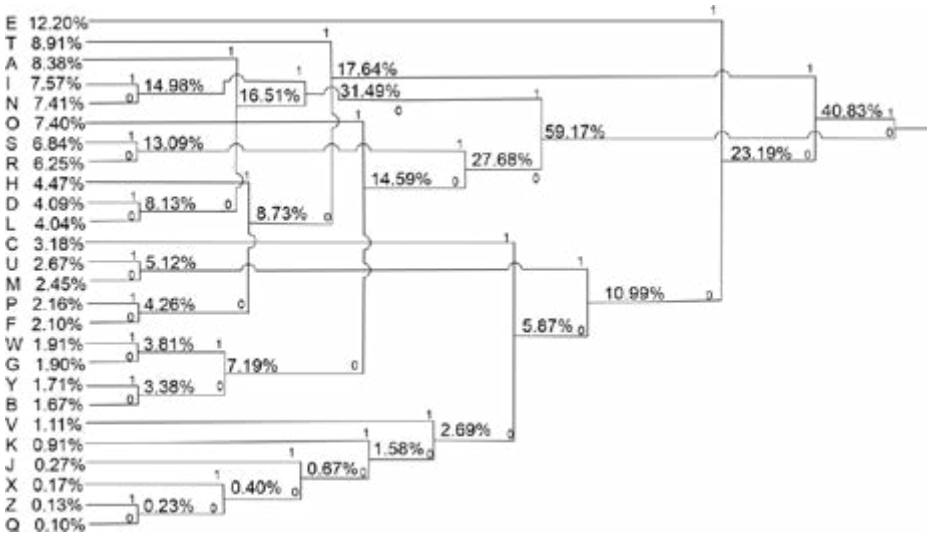
$$\bar{L} = 10 \cdot 0.23\% + 9 \cdot 0.17\% + 8 \cdot 0.27\% + 7 \cdot 0.91\% + \\ + 6 \cdot 8.30\% + 5 \cdot 20.29\% + 4 \cdot 48.12\% + 3 \cdot 21.11\% = 4.1942 \text{ bita / simbolu.}$$

A	8.38%	B	1.67%	C	3.18%	D	4.09%	E	12.20%	F	2.10%
G	1.90%	H	4.47%	I	7.57%	J	0.27%	K	0.91%	L	4.04%
M	2.45%	N	7.41%	O	7.40%	P	2.16%	Q	0.10%	R	6.25%
S	6.84%	T	8.91%	U	2.67%	V	1.11%	W	1.91%	X	0.17%
Y	1.71%	Z	0.13%								

Tabela III.16. Vjerovatnoće simbola engleskog alfabeta u izvoru (u pitanju su dvije knjige)

A	0101	B	000000	C	10001	D	01001
E	101	F	11000	G	000010	H	1101
I	0111	J	10000001	K	1000001	L	01000
M	00010	N	0110	O	0001	P	11001
Q	1000000000	R	0010	S	0011	T	111
U	00011	V	100001	W	000011	X	100000001
Y	000001	Z	1000000001				

Tabela III.17. Hafmenov kod za engleski alfabet iz primjera



Slika III.17. Hafmenovo stablo za engleski alfabet iz primjera

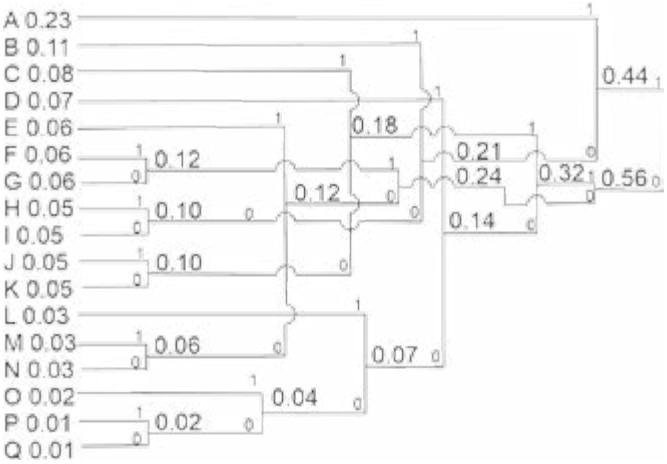
3.19. Odrediti Hafmenov kod i prosječnu dužinu kodne riječi ako su vjerovatnoće pojedinih simbola 0.23, 0.11, 0.08, 0.07, 0.06, 0.06, 0.06, 0.05, 0.05, 0.05, 0.05, 0.03, 0.03, 0.03, 0.02, 0.01, 0.01. Ponoviti proceduru ako se Hafmenovo kodiranje primijeni s fiksnom dužinom kodne riječi za 8 najmanje vjerovatnih simbola.

Rješenje: Hafmenova stabla za dva tražena rješenja su data na Slikama III.18 i III.19. Sami rekonstruišite kodne riječi. Prosječna dužina kodne riječi u prvom slučaju je:

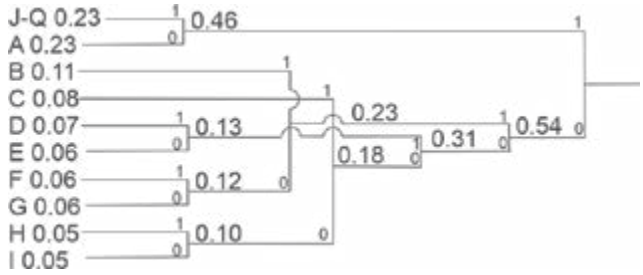
$$\bar{L}_1 = 0.02 \cdot 7 + 0.02 \cdot 6 + 0.19 \cdot 5 + 0.43 \cdot 4 + 0.11 \cdot 3 + 0.23 \cdot 2 = 3.72 \text{ bita/simbolu.}$$

Entropija ovoga sistema je $H(X) \approx 3.6941$ bita što ukazuje na činjenicu da je Hafmenov kod veoma blizak ovoj vrijednosti. U slučaju koda gdje je osam najrjeđih simbola kodirano kodom fiksne dužine, imamo prosječnu dužinu:

$$\bar{L}_2 = 0.10 \cdot 5 + 0.33 \cdot 4 + 0.11 \cdot 3 + 0.23 \cdot 2 + 0.23 \cdot 5 = 3.76 \text{ bita/simbolu.}$$



Slika III.18. Hafmenovo stablo za alfabet dat u zadatku 3.19



Slika III.19. Hafmenovo stablo za alfabet dat u zadatku 3.19, gdje je 8 simbola J−Q, koji se pojavljuju najrjeđe, kodirano fiksnom dužinom

Dakle, predmetnim kodiranjem smo samo za 0.04 bita/simbolu povećali prosječnu dužinu kodne riječi, ali uz korist da je najduža riječ ograničena na pet bita umjesto na sedam, u slučaju punog kodiranja. Osam najrjeđih simbola, koji su u kodnom stablu prikazani kao jedna vrijednost, mogu se kodirati petobitno prefiksom koji daje stablo (11) i sufiksom od tri bita:

J	11000	*K*	11001	*L*	11010	*M*	11011
N	11100	*O*	11101	*P*	11110	*Q*	11111.

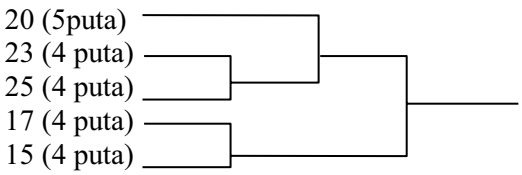
3.20. Posmatrajte funkciju $y(n) = \text{round}(20 + 5\sin(0.2\pi n))$ gdje funkcija round znači zaokruživanje na najbliži cijeli broj. Vrijednosti $n \in [0,20]$. Kreirati Hafmenov kod za datu sekvencu i kombinaciju diferencijalni i Hafmenov kod. Odrediti prosječne dužine kodne riječi u oba slučaja.

Rješenje: Sekvenca se sastoji od 21 vrijednosti, datih redom:

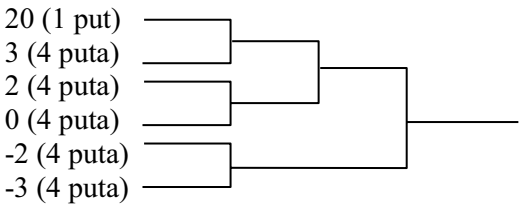
20	23	25	25	23	20	17
15	15	17	20	23	25	25
23	20	17	15	15	17	20.

Hafmenovo stablo je prikazano na Slici III.20 (nisu upisani biti koji se dobijaju prilikom kodiranja). Simbol koji se pojavljuje najčešće kodiran je sa 2 bita, kao i vrijednosti 17 i 15, dok su ostali kodirani sa tri bita. Dakle, potrebno nam je ukupno (ovdje ponovo ne razmatramo složenost prenosa tabele sa kodovima sa predajnika prema prijemniku) $13 \times 2 + 8 \times 3 = 50$ bita za kodiranje ove poruke. Primijenimo sada diferencijalni kod:

20	3	2	0	−2	−3	−3
−2	0	2	3	3	2	0
−2	−3	−3	−2	0	2	3.



Slika III.20. Hafmenovo stablo za kodiranje poruke iz zadatka 3.20



Slika III.21. Hafmenovo stablo za kodiranje poruke iz zadatka 3.20 nakon primjene diferencijalnog kodiranja

Na Slici III.21 prikazano je diferencijalno kodiranje ove poruke. Simboli -2 i -3 su kodirani sa 2 bita, dok su ostali kodirani sa tri bita, pa nam je u ovom slučaju potrebno $8\times 2+13\times 3=55$ bita, što je više nego u slučaju direktne primjene Hafmenovog kodiranja. Zbog čega se dešava ova devijacija?

3.21. Šalju se $n=5$ i.i.d. simbola sa vjerovatnoćom jedinice $p=0.7$ i nule $q=0.3$. Kreirati Hafmenov kod za dati skup riječi izvornog alfabeta i odrediti prosječnu dužinu kodne riječi.

Rješenje: U Tabeli III.18 prikazane su petobitne i.i.d. sekvence, njihove vjerovatnoće i kodovi koje smo formirali. Sami rekonstruišite odgovarajuće stablo. Naravno, u zavisnosti od postupka formiranja kodnog stabla i dodjele kodiranja mogući su i drugi oblici kodnih riječi. Prosječna dužina kojom se kodira ova sekvenca je:

$$\overline{L}_5 = 3\times(0.16897+0.07203)+4\times 4\times 0.07203+\ 5\times(9\times 0.03087+0.01323)+$$
$$6\times(0.03087+9\times 0.01323)+7\times 2\times 0.00567+8\times\ (3\times 0.00567+0.00243)=$$
$$=0.72300+1.15248+1.45530+0.89964+0.07938+0.15552=4.4653.$$

Ovo daje prosječnu dužinu kodne riječi po jednom simbolu od 0.8931 bita/simbolu što je tek nešto preko entropije od $H(X)\approx 0.8813$ bita.

Riječ	Vjerovatnoća	Kod	Riječ	Vjerovatnoća	Kod
11111	0.16807	111	11110	0.07203	1101
11101	0.07203	1100	11011	0.07203	1011
10111	0.07203	1010	01111	0.07203	011
11100	0.03807	10001	11010	0.03807	10011
11001	0.03807	00111	10110	0.03807	00110
10101	0.03807	00101	10011	0.03807	00100
01110	0.03807	00011	01101	0.03807	00010
01011	0.03807	00001	00111	0.03807	100001
11000	0.01323	000001	10100	0.01323	000000
10010	0.01323	010111	10001	0.01323	010110
01100	0.01323	010101	01010	0.01323	010100
01001	0.01323	010011	00110	0.01323	010010
00101	0.01323	010001	00011	0.01323	010011
10000	0.00567	0100001	01000	0.00567	010000
00100	0.00567	10000011	00010	0.00567	1000010
00001	0.00567	10000001	00000	0.00243	1000000

Tabela III.18. Petobitne i.i.d. sekvence kodirane Hafmenovim kodom

3.22. Posmatrajte dijagram stanja Markovljevog procesa drugog reda koji je dat prethodno iz Primjera II.3 i II.5. Primjenom Hafmenovog kodiranja kodirati sekvencu 000011111110.

Rješenje: Podsjetimo se da smo imali Markovljev sistem drugog reda s uslovnim vjerovatnoćama $P(0|00)=P(1|11)=0.7$, $P(1|00)=P(0|11)=0.3$, $P(0|01)=P(0|10)=P(1|01)=P(1|10)=0.5$, dok su združene vjerovatnoće u stacionarnom stanju $p(00)=p(11)=5/16$, dok je $p(10)=p(01)=3/16$, odnosno $p(0)=1/2$ i $p(1)=1/2$. Dakle, ako bismo kodirali pojedinačne bite na osnovu vjerovatnoća u stacionarnom režimu, ne bismo ostvarili popravku jer bi kodiranje bilo sa jednim bitom. Isto bi bilo u slučaju kodiranja po dva bita, korišćenjem vjerovatnoća u stacionarnom režimu (u prvom koraku bismo obuhvatili dva najmanje vjerovatna simbola 3/16 + 3/16, a zatim simbole sa vjerovatnoćama 5/16 i 5/16, čime bi se ovi parovi kodirali sa po 2 bita po paru, odnosno 1 bit po ulaznom bitu). Postoji više načina da se ipak uštedi nešto i u ovom slučaju. Prvi način je da kodiramo trojke simbola sa vjerovatnoćama:

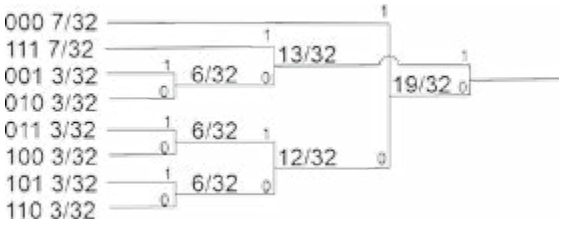
$P(000)=P(0|00)P(00)=7/32$ $\quad P(111)=7/32$
$P(001)=3/32$ $\quad P(110)=3/32$
$P(010)=3/32$ $\quad P(101)=3/32$
$P(100)=3/32$ $\quad P(011)=3/32.$

Na Slici III.22 prikazano je Hafmenovo stablo koje korespondira predmetnim trojkama. Prosječna dužina kodne riječi za predstavljanje trojke bita je:

$$\bar{L}_3 = 2\cdot\frac{14}{32}+3\cdot\frac{6}{32}+4\cdot\frac{12}{32}=\frac{94}{32}=\frac{47}{16}=2.9375 \text{ bita},$$

dok je po pojedinačnom bitu $\bar{L}=\bar{L}_3/3=0.9792$ bita. Očigledno je postignuta ušteda (premda mala). Poruka se kodira tako što se izdvajaju po tri bita i kodiraju u skladu sa Hafmenovim stablom (radi lakše preglednosti izdvojili smo kodove koji kodiraju pojedine trojke):

01 0011 11 0000.



Slika III.22. Hafmenovo stablo za kodiranje trobita kod Markovljevog sistema II reda

Nije ovo jedini način da se izvrši kodiranje u predmetnom slučaju. Naime, možemo da posmatramo situaciju da znamo prethodna dva bita, na primjer 00, i da se pitamo kolika je vjerovatnoća pojave bilo koje kombinacije dvobita nakon 00:

$$P(00|00)=P(0|00)P(0|00)=0.49 \qquad P(01|00)=0.15$$
$$P(10|00)=P(1|00)P(0|00)=0.21 \qquad P(11|00)=0.15.$$

Slično se dobija kada imamo kombinaciju 11:

$$P(11|11)=P(1|11)P(1|11)=0.49 \qquad P(01|11)=0.21$$
$$P(10|00)=0.15 \qquad P(00|11)=0.15.$$

Ponovimo postupak za kombinacije 01 i 10:

$$P(00|01)=P(0|01)P(0|00)=0.35 \qquad P(01|01)=0.25$$
$$P(10|01)=0.15 \qquad P(11|01)=0.25$$
$$P(00|10)=0.25 \qquad P(01|10)=0.15$$
$$P(10|10)=0.25 \qquad P(11|10)=0.35.$$

Na Slici III.23 prikazani su Hafmenovi kodovi, pod uslovom da su dvobiti prethodnici 00 i 11. U slučaju prethodnika 01 i 10 ne ostvaruje se ušteda Hafmenovim kodiranjem, pa ga nećemo ni vršiti. Za kombinacije 00 i 11, kao prethodnika, prosječne dužine kodne riječi su:

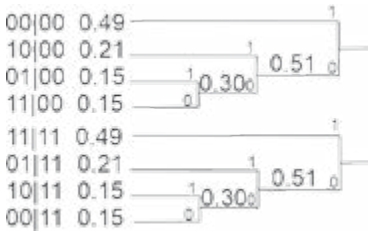
$$\bar{L}_{00}=\bar{L}_{11}=1\cdot 0.49+2\cdot 0.21+3\cdot 0.30=1.82,$$

ili po bitu $\bar{L}_{00}/2=\bar{L}_{11}/2=0.91$. Ukupna prosječna dužina kodne riječi sada je:

$$\bar{L}=\frac{5}{16}\frac{\bar{L}_{00}}{2}+\frac{5}{16}\frac{\bar{L}_{11}}{2}+\frac{3}{16}+\frac{3}{16}\approx 0.9437 \text{ bita}.$$

Ovo je očigledno nešto povoljnije nego što je prva varijanta. Izvršimo kodiranje tako da direktno kodiramo prva dva bita, kao i svaka dva bita nakon kombinacija 01 ili 10, a da ostale kodiramo u skladu sa Slikom III.23 (ponovo odvajamo dvobite da bismo lakše pratili):

00 1 000 1 1 001.



Slika III.23. Hafmenova stabla za kodiranje uslovno dvobitnih sekvenci nakon prethodne dvobitne sekvence

Razmotrite eventualno druge varijante kodiranja i odredite graničnu vrijednost prosječne dužine kodne riječi u slučaju predmetnog Markovljevog sistema II reda.

3.23. Dat je string:

abacbabcbabcbababcbacbbabaaabbcbabac,

koga treba kodirati LZ kodom. Rječnik ima 16 upisa i na početku sadrži kod izvora. Nakon popunjavanja, rječnik se resetuje na stanje sa kodom izvora.

Rješenje: Premda rječnik ima 16 upisa, kodiraćemo ga zapisujući prefikse u obliku rednih brojeva. U Tabeli III.19 prikazali smo rječnik (prva tri upisa predstavljaju kod izvora). Kodiranje je dato kao:

ab ac ba bc bab cb aba bcb acb baba aa bb cba

(1,*b*), (1,*c*), (2,*a*), (2,*c*), (6,*b*), (3,*b*), (4,*a*), (7,*b*), (5,*b*), (8,*a*), (1,*a*), (2,*b*), (9,*a*), (6,*c*).

Uočimo da je tek u posljednjem koraku prepunjen rječnik, čime se stanje u njemu resetuje (vraća na početno).

3.24. Kodiranje se obavlja u ternarnom kodu *a*, *b*, *c*. Odrediti LZ kod sekvence:

ab aba ba cb abc bb cc aa ab bc cb ab ac.

a) Pretpostaviti da je rječnik na početku prazan i da se nakon 8 upisa rječnik resetuje, odnosno prazni.

b) Postoji rječnik koji posjeduje simbole *a*, *b*, *c*. Rječnik se ne resetuje u okviru ove sekvence. Zadatak odraditi korak po korak, sa detaljnim opisom rječnika.

Rješenje: a) Pretpostavimo da je rječnik na početku prazan. Kodiramo simbol *a*, a u rječnik upisujemo na poziciji:

001 *a* do sada kodirani dio sekvence je 000*a*.

Prefiks	Sadržaj	Prefiks	Sadržaj	Prefiks	Sadržaj
1	*a*	8	*bab*	15	*bb*
2	*b*	9	*cb*	16	*bba*
3	*c*	10	*aba*	1	*a*
4	*ab*	11	*bcb*	2	*b*
5	*ac*	12	*acb*	3	*c*
6	*ba*	13	*baba*		
7	*bc*	14	*aa*		

Tabela III.19. Rječnik kod LZ koda iz zadatka 3.23

Zatim kodiramo *b* koje do sada ne postoji u rječniku, i u rječnik upisujemo:

010 *b* a kod je je 000*b*.

Zatim kodiramo *ab* (*a* već postoji u rječniku, dok je *b* sufiks). Upis u rječnik je:

011 *ab* a kod je 001*b*.

Zatim se kodira *aba* (*ab* je prefiks, dok je *a* sufiks):

100 *aba* kod je 011*a*.

Zatim se kodira *c*, koje se do sada nije pojavljivalo u sekvenci:

101 *c* kod je 000*c*.

Naredna kombinacija je *ba*, koja se upisuje u rječnik na poziciji 6:

110 *ba* i kodira kao 010*a*.

Sljedeća kombinacija je *bc*, koja se upisuje u rječnik na poziciji 7:

111 *bc* i kodira kao 010*c*.

Naredna kombinacija je *bb*, koja se upisuje u prazan rječnik na poziciji 001, a sastoji se od prethodnog *b* i novododatog *b*:

001 *bb* kodira se 010*b* (nakon ovoga je prazno ono što je bilo upisano u rječniku, može se eventualno početi i od kodiranja samog *b*, ali smo na ovaj način uštedjeli nekoliko bita u kodiranju).

Sljedeća kombinacija je *c* i upisuje se u rječnik na poziciji 010:

010 *c* 000*c*.

Nakon nje slijedi *cc*:

011 *cc* 010*c*.

Zatim je kombinacija *a*, pa za njom slijedi *ab*:

100 *a* 000*a*
101 *ab* 100*b*.

Nakon ove kombinacije ide *b* (uočite da je *bb* već u rječniku), pa zatim *ccb*:

110 *b* 000*b*
111 *ccb* 011*b*.

U rječniku koji treba isprazniti imamo već kombinaciju koja slijedi *ab*, pa nije pametno da je ne iskoristimo. Stoga kodiramo *aba*, pa tek onda, zajedno s upi-

sivanjem ovog elementa u rječnik (upisivanje nije obavezno), vršimo pražnjenje rječnika i kodiranje preostalog simbola *c*:

000 *aba* 101*a*
001 *c* 000*c*.

b) Kada u rječniku na početku imamo kôd izvora, na lokacijama 000, 001 i 010 su redom upisani *a*, *b* i *c*. Nakon toga, kodiramo redom: *ab*, *aba*, *ba*, *cb* i *abc*:

011 *ab* 000*b*
100 *aba* 011*a*
101 *ba* 001*a*
110 *cb* 010*b*
111 *abc* 011.

Sada možemo isprazniti ostatak rječnika i kodirati redom: *bb*, *cc*, *aa*, *ab* i *bc*:

011 *bb* 001*b*
100 *cc* 010*c*
101 *aa* 000*a*
110 *ab* 000*b*
111 *bc* 001*c*.

Konačno, preostaje nam kodiranje *cb*, *ab*, *ac*:

011 *cb* 010*b*
100 *ab* 000*b*
101 *ac* 000*c*.

U našoj realizaciji drugi algoritam je nešto jednostavniji pošto nam je bilo potrebno samo 13 upisa, dok je prvi zahtijevao 15 upisa. U realnim aplikacijama prednost jednog ili drugog postupka će se osjetiti tek nakon dužeg stringa, a ujedno bitno zavisi i od složenosti punjenja i pražnjenja rječnika, jer ako startujemo s rječnikom u kojem ima nešto (recimo kod izvora), brže ćemo rječnik puniti i češće prazniti.

3.25. Izvršiti aritmetičko kodiranje ternarnog koda sa simbolima (*a*,*b*,*c*). Terminacioni simbol je #. Vjerovatnoća terminacionog simbola za prvi karakter je 0.1, a za svaki naredni raste za po 0.05. Za prvi karakter se može pretpostaviti da su vjerovatnoće (*a*,*b*,*c*) iste. Za svaki naredni se vrši ažuriranje na osnovu broja karaktera koji su se do sada pojavili u sekvenci (koristiti Laplasovo pravilo za ažuriranje). Kodirati poruku *abbbbacc*#.

Rješenje: Kod prvog simbola, vjerovatnoće su *a* u granicama [0,0.3), *b* u granicama [0.3,0.6), *c* u granicama [0.6,0.9), a # [0.9,1). Pošto je poslato *a*, onda imamo granice [0,0.3).

Prilikom kodiranja narednog simbola, vjerovatnoća simbola # je 0.15 ($P(\#|a)=0.15$), dok se ostali raspoređuju $P(a|a)=0.5$, $P(b|a)=P(c|a)=0.25$. Dakle, sada su granice: za a je granica $[0,0.3\times 0.5\times 0.85)=[0,0.1275)$, dok je simbol b u granicama $[0.1275,0.1275+0.3\times 0.25\times 0.85)=$ $[0.1275,0.19125)$, simbol c je u granicama $[0.19125,\ 0.19125+0.3\times 0.25\times 0.85)=[0.19125,0.255)$, konačno, terminacioni simbol je u granicama $[0.255,0.3)$. Kako smo poslali simbol b izabran je interval $[0.1275,0.19125)$.

Prilikom kodiranja narednog simbola, vjerovatnoća terminacionog simbola je 0.2, dok su vjerovatnoće simbola a i b $P(a|ab)=P(b|ab)=2/5=0.4$, dok je vjerovatnoća $P(c|ab)=0.2$. Razmak između gornje i donje granice intervala je 0.06375. Kako je ponovo izabran simbol b, dobili smo da su nove granice u intervalu: $[0.1275$ $+0.06375\times 0.8\times 0.4,0.1275+0.06375\times 0.8\times 0.8)=[0.1479,0.1683)$. Sveli smo granice na interval širine 0.0204.

Vjerovatnoća terminacionog simbola je sada 0.25, dok su vjerovatnoće simbola a, b i c, redom: 1/3, 1/2 i 1/6. Ponovo se šalje b, pa je selektovan interval $[0.1479+0.0204\times 0.75/3,\ 0.1479+0.0204\times 0.75\times 5/6)=[0.153,0.16065)$. Širina intervala je 0.00765.

Sada vjerovatnoća terminacionog simbola raste do 0.3. Vjerovatnoće simbola alfabeta su sada, redom: 2/7, 4/7 i 1/7. Ponovo se šalje b simbol, pa su granice intervala:

$[0.153+0.00765\times 0.7\times 2/7,0.153+0.00765\times 0.7\times 6/7)=[0.15453,0.15759)$. Širina intervala je 0.00306.

Vjerovatnoća terminacionog simbola sada raste do 0.35. Vjerovatnoće simbola alfabeta su 1/4, 5/8 i 1/8. Pošto je poslato a, dobijamo interval:

$[0.15453,\ 0.15453\times 0.65\times 0.00306/4)=[0.15453,0.15502725)$. Širina intervala je svedena na 0.00049725.

Prilikom kodiranja narednog simbola, vjerovatnoća terminacionog karaktera je 0.4, dok su vjerovatnoće pojedinih simbola 3/9, 5/9 i 1/9. Pošto je selektovan c, granice intervala su:

$[0.15453+0.00049725\times 0.6\times 8/9,\ 0.15453+0.00049725\times 0.6)=[0.1547952,$ $0.15482835)$. Širina intervala je sada: 0.00003315.

Prilikom kodiranja narednog simbola, vjerovatnoća terminacionog karaktera je 0.45, dok su vjerovatnoće ostalih simbola 0.3, 0.5 i 0.2. Pošto je poslato c, onda su granice intervala:

$[0.1547952+0.00003315\times 0.55\times 0.8,\ 0.1547952+0.00003315\times 0.55)=$
$=[0.154809786,\ 0.1548134325)$. Širina intervala je svedena na: 0.0000036465.

Konačno, u posljednjem koraku kodira se sam terminacioni karakter, koji se pojavljuje sa vjerovatnoćom 0.5, pa su granice intervala (gornja se ne mijenja):

$[0.154809786 + 0.0000036465/2,\quad 0.1548134325) = [0.15481160925, 0.1548134325)$.

Širina intervala je svedena na: 0.00000182325.

Obje granice se pretvaraju u binarne brojeve čijih je prvih 17 decimala 00100111101000011. Treba uočiti da naredne tri cifre 100 sigurno pripadaju ovom intervalu, pa je dovoljno priključiti prvu cifru:

001001111010000111.

3.26. Izvršiti aritmetičko kodiranje binarnog koda sa simbolima (0,1). Terminacioni simbol je #. Vjerovatnoća terminacionog simbola za prvi karakter je 0.1, a za svaki naredni raste za po 0.05. Za prvi karakter se može pretpostaviti da su vjerovatnoće karaktera (0,1) iste. Za svaki naredni se vrši ažuriranje na osnovu broja karaktera koji su se do sada pojavili u sekvenci (koristiti Laplasovo pravilo). Kodirati poruku 011110001#.

Rješenje: Prvi simbol koji je poslat je 0. Pretpostavimo da to redukuje interval na zonu [0,0.45). Vjerovatnoća terminacionog karaktera se sada penje na 0.15, a vjerovatnoće simbola '0' i '1' su respektivno: $(1-0.15)\times 2/3 = 0.5667$ i $(1-0.15)/3 = 0.2833$. Kako je poslata jedinica, tako su se granice intervala suzile na:

$[0 + 0.5667 \times 0.45, 0 + 0.85 \times 0.45) = [0.2550, 0.3825)$ širina intervala je 0.1275.

Vjerovatnoća terminacionog karaktera je sada 0.2, a po 0.4 su vjerovatnoće '0' i '1' simbola:

$[0.2550 + 0.4 \times 0.1275, 0.2550 + 0.8 \times 0.1275) = [0.3060, 0.3570)$, a širina intervala je redukovana na 0.051.

Za naredni simbol vjerovatnoća terminacionog karaktera raste na 0.25, a vjerovatnoće simbola '0' i '1' su: 0.25 i 0.50, respektivno. Pošto šaljemo simbol '1', odabrali smo interval:

$[0.3060 + 0.0510 \times 0.25, 0.3060 + 0.0510 \times 0.75) = [0.31875, 0.34425)$; širina intervala je 0.0255.

Za naredni simbol vjerovatnoća pojavljivanja terminacionog karaktera raste na 0.30, a vjerovatnoće simbola '0' i '1' su respektivno 0.175 i 0.525, a pošto šaljemo '1', dobijamo:

[0.3232125, 0.3366), dok se širina intervala smanjuje na 0.0133875.

U ovom koraku je vjerovatnoća terminacionog simbola porasla na 0.35, dok su vjerovatnoće simbola ‘0’ i ‘1’ respektivno 0.13 i 0.52. Pošto šaljemo 0, imamo interval sveden na:

[0.3232125, 0.324952875), koji je sužen na 0.001740375.

Sada je vjerovatnoća terminacionog simbola porasla na 0.4, a vjerovatnoće ‘0’ i ‘1’ su redom: 0.2 i 0.4. Pošto se šalje simbol ‘0’, selektovali smo prvi interval:

[0.3232125, 0.323560575), širine $3.48075 \cdot 10^{-4}$.

U narednom koraku vjerovatnoća terminacionog simbola raste na 0.45. Vjerovatnoće simbola ‘0’ i ‘1’ su respektivno 33/140 i 44/140. Šalje se 0, a to znači da smo selektovali interval

[0.3232125, 0.32329454625), koji je sužen na $8.2046 \cdot 10^{-5}$.

U narednom koraku vjerovatnoća terminacionog simbola raste na 0.5. Vjerovatnoće simbola ‘0’ i ‘1’ su iste i iznose 0.25. Pošto se šalje simbol ‘1’, to znači da je selektovan interval

[0.3232330115625, 0.323253523125), čija je širina $2.05116 \cdot 10^{-5}$.

Konačno, u posljednjem intervalu šaljemo terminacioni simbol koji se pojavljuje sa vjerovatnoćom 0.55, tako da smo selektovali interval:

[0.32324429292188, 0.323253523125), čime je interval sužen na $1.1281 \cdot 10^{-5}$.

Što se tiče binarne poruke koja se treba poslati, uočljivo je da su prvih 16 bita koji kodiraju početak intervala isti kao 16 bita koji kodiraju kraj intervala, pa ih šaljemo u kanal. Nakon toga imamo 0 za početak i 1 za kraj intervala, što nije dovoljno da donesemo odluku o konkretnom slanju, ali nakon dva bita 00 i 10 možemo zaključiti da je kombinacija 01 sigurno u intervalu, tako da nju možemo poslati da zaključimo kodnu sekvencu:

početak intervala	[0101001011000000 0010]
kraj intervala	[0101001011000000 1011]
poslata sekvenca	[0101001011000000 01].

3.27. Primjenom aritmetičkog kodiranja primljena je poruka 0.30. Smatrati da se alfabet sastoji od simbola $\{a, b, c, d\}$ i terminacionag simbola #. Vjerovatnoće pojavljivanja simbola su 15%, 25%, 30%, 10% i 20%. Dekodirati primljenu poruku. Smatrati da se vjerovatnoće simbola ne mijenjaju.

Rješenje: Na početku simbolu '*a*' odgovara interval od [0,0.15), a simbolu '*b*' interval [0.15,0.40). Zaključujemo da je sigurno poslato '*b*'. Interval je sužen na $p = 0.4 - 0.15 = 0.25$. Odredimo sada podintervale:

za '*a*' $[0.15, 0.15 + 0.25 \times 0.15) = [0.15, 0.1875)$
za '*b*' $[0.1875, 0.15 + 0.25 \times 0.40) = [0.1875, 0.25)$
za '*c*' $[0.25, 0.15 + 0.25 \times 0.7) = [0.25, 0.325)$.

Ne moramo dalje da idemo. Sada smo sigurni da je poslije '*b*' poslato '*c*'. Interval je sužen na 0.075. Posmatrajmo slanje sljedećeg simbola:

za '*a*' $[0.25, 0.25 + 0.075 \times 0.15) \approx [0.25, 0.2612)$
za '*b*' $[0.2612, 0.25 + 0.075 \times 0.4) \approx [0.2612, 0.28)$
za '*c*' $[0.28, 0.25 + 0.075 \times 0.7) \approx [0.28, 0.3025)$.

Dakle, treći simbol je takođe 'c'. Interval je sužen na 0.0225. Sada posmatrajmo šta se dešava s terminacionim karakterom:

za '#' $[0.28 + 0.0225 \times 0.8, 0.3025) = [0.298, 0.3025)$.

Više nego lako zaključujemo da je posljednji simbol koji se šalje terminacioni, što znači da je poslata sekvenca *bbc*#.

III.8.2 Softverska realizacija

A. Dugačak niz karaktera smješten je u vektoru B. Procijeniti entropiju alfabeta (u pitanju je tekst na engleskom jeziku koji se sastoji samo od malih slova). Zatim odrediti združenu entropiju dva uzastopna simbola.

```
>> Total=zeros(1,26);
>> for k=1:length(B)
Total(B(k)-'a'+1)=Total(B(k)-'a'+1)+1;
end
>> pest=Total/sum(Total);
>> H=- sum(-pest.*log2(pest)
```

Sada dajemo dio koda za određivanje entropije dva simbola. Matrica Y broji koliko se puta pojavio neki simbol iza drugog simbola. Npr. Y(1,1) predstavlja koliko se puta pojavio simbol 'a' iza simbola 'a', dok Y(1,2) označava koliko puta se pojavio simbol 'b' nakon simbola 'a'. Ovo bi trebalo u konkretnom primjeru da bude proporcionalno uslovnoj vjerovatnoći p('b'|'a').

```
>> Y=zeros(26,26);
>> for k=1:length(B)-1
Y(B(k)-'a'+1,B(k+1)-'a'+1)=Y(B(k)-'a'+1,B(k+1)-'a'+1)+1;
end
```

```
>> pest2=Y/sum(sum(Y));
>> H1=sum(-(pest2(:)+eps).*log2((pest2(:)+eps)))/2
```

Malu vrijednost $\text{eps} = 2^{-52}$ dodavali smo na vjerovatnoće da bismo izbjegli logaritam od 0 jer se dešava da se neke kombinacije ne dogode nijednom.

B. Kreirajte funkcije za RLE kodiranje i dekodiranje:

```
function y=rle_kod(x)
N=length(x);
y(1)=x(1);
y(2)=1;
l=1;
for k=2:N
  k
  if(x(k)==x(k-1)),y(l+1)=y(l+1)+1;
else,l=l+2;y(l)=x(k);y(l+1)=1;end
end
```

Uzimamo da je prvi odbirak element koda, a da je drugi 1. Ovaj drugi predstavlja broj pojavljivanja prvog elementa. Ako je naredni element isti kao prethodni x(k)==x(k-1), uvećavamo y(l+1)=y(l+1)+1, a ako nije, prelazimo na novi par, pa prvi element postavljamo da je x(k), a drugi da je 1. Drugi element ponovo uvećavamo za svako ponavljanje elementa u polaznom nizu. Dekodiranje se može obaviti na još jednostavniji način, na primjer:

```
function x=rle_dec(y)
N=length(y);
x=zeros(1,length(sum(y(2:2:N)));
ind=0;
for k=1:2:N
x(ind+1:ind+y(k+1))=y(k)*ones(1,y(k+1));
ind=ind+y(k+1);
end
```

Uzimamo parove elemenata iz niza y i zatim prebacujemo u niz x podnizove s odgovarajućim elementom odgovarajuće dužine y(k)*ones(1,y(k+1)).

C. Realizujte Hafmenovo kodiranje.

Pretraživanjem po internetu može se naći veliki broj različitih realizacija Hafmenovog kodiranja. Hafmenov kod ima pogodnu strukturu stabla (nadam se da ovo nije potrebno objašnjavati), pa je zgodan da se realizuje putem koncepata objektno orijentisanog programiranja. Naime, kodno stablo se može realizovati kao struktura drveta sa dva pokazivača koji pokazuju na lijevog i na desnog sina. Međutim, pošto se naše stablo mora kreirati unazad, od čvorova ka korijenu, potreban je još jedan

pokazivač koji pamti roditelja. Pošto se dekodiranje obavlja od glave ka listovima, zgodno je da svaki čvor pamti i glavu kao statički podatak član, zajednički za sve čvorove u sistemu. Dobra strana objektno orijentisanog pristupa je i činjenica da se kreiranje novih čvorova može obaviti u konstruktoru, te da se uništavanje čvorova može obaviti u destruktoru. Ovim se ne opterećuje ostatak koda provjerama postojanja čvora, odnosno uništavanjem čvorova i njihovim kreiranjem. Mnogi alati, srećom, posjeduju ugrađene naredbe za Hafmenovo kodiranje i dekodiranje. U nastavku dajemo nekoliko osnovnih naredbi, putem kojih se u programskom paketu MATLAB mogu vršiti ove operacije:

```
>> symbols = [1:6];
>> p = [.5 .125 .125 .125 .0625 .0625];
>> [dict,avglen] = huffmandict(symbols,p);
>> dict{3,2}
>> avglen
>> actualsig = randsrc(1,100,[symbols; p]);
>> comp = huffmanenco(actualsig,dict);
>> dsig = huffmandeco(comp,dict);
>> isequal(actualsig,dsig)
```

Promjenljiva symbols predstavlja pojedine simbole koda (u ovom slučaju 1, 2, 3, 4, 5, 6), dok su u vektoru p postavljene vjerovatnoće pojedinih simbola. Funkcija huffmandict formira kodno stablo (dict) i rječnik podataka koji predstavlja ćelije dimenzija 6×2, gdje je 6 broj simbola u našem ulaznom alfabetu. Na primjer, dict{3,1} predstavlja treći kodirani simbol (u primjeru je to 3), dict{3,2} daje kodnu riječ kojom je kodiran (u primjeru je to 111), avglen je prosječna dužina kodne riječi. Funkcija randsrc, u predstavljenom primjeru, ima tri argumenta, prva dva su dimenzije polja slučajnih brojeva, koje se kreira naredbom (u ovom slučaju 1×100), dok je treći argument matrica čija prva vrsta predstavlja simbole ulaznog alfabeta, a druga vjerovatnoću pojave tih simbola. Dakle, actualsig predstavlja niz simbola ulaznog alfabeta formiranog u skladu sa zadatom distribucijom vjerovatnoća alfabeta; huffmanenco(actualsig,dict) vrši kodiranje sekvence actualsig putem rječnika dict. Dekodiranje binarne sekvence deco Hafmenovim dekoderom, na osnovu rječnika dict, obavlja se sa huffmandeco(comp,dict). Posljednja naredba provjerava da li su originalna i dekodirana sekvenca iste i pošto je rezultat 1, zaključujemo da jesu.

D. U trenutku pisanja ove knjige ne postoji ugrađena MATLAB funkcija niti toolbox koji je realizovao LZ/LZW kodiranje. Srećom, na sajtu za razmjenu MATLAB fajlova (https://www.mathworks.com/matlabcentral/fileexchange/) postoji nekoliko alata koji se mogu koristiti za ove potrebe. Na primjer, funkcije norm2lzw i lzw2norm autora Đuzepea Ridina (ital. *Giuseppe Ridino*). Radi familijarizacije, napravljen je i demonstracioni program (demo) koji se može

iskoristiti za testiranje ovih funkcija (napominjem da su riječi razdvojene specijalnim simbolom \, te da je pretpostavljeno kodiranje sa predefinisanom ASCII tabelom, koja je smještena u rječnik). Predmetne funkcije su unaprijeđene od strane Dankana Barklija (engl. *Duncan Barclay*) i dostupne su na istom sajtu. Konačno, postoji i funkcija lempel_ziv, koju je kreirao Andrea Ćirilo (ital. *Andrea Cirillo*), a koja je namijenjena za LZ kodiranje. U svakom slučaju, za programera ne bi trebalo da predstavlja nepremostiv problem da na osnovu ovih kodova ili sopstvenog znanja napravi verzije funkcija za LZW kodiranje i dekodiranje.

E. MATLAB posjeduje funkcije za aritmetičko kodiranje i dekodiranje za slučaj kodova kod kojih je vjerovatnoća simbola unaprijed poznata i ne mijenja se tokom kodne riječi. Kreirajmo niz od 400 trojki i po 50 jedinica i dvojki:

```
>> seq = repmat([3 3 1 3 3 3 3 3 2 3],1,50);
```

Pretpostavimo da unaprijed znamo tačan procenat pojedinih simbola (ovdje je to lako, ali u praksi ne mora da bude tako):

```
>> counts = [10 10 80];
```

Funkcija arithnenco se koristi da bi izvršila kodiranje:

```
>> code = arithenco(seq,counts);
```

dok se dekodiranje obavlja naredbom rithdeco:

```
>> dseq = arithdeco(code,counts,length(seq));
```

U predmetnom primjeru dužina kodne riječi je 470, a dobija se sa:

```
>> length(code)
>> length(dseq)
```

što je ušteda od 6% u odnosu na 500 simbola originalne poruke. U slučaju da imamo pogrešnu pretpostavku o vjerovatnoćama simbola:

```
>> counts = [10 15 75];
>> code = arithenco(seq,counts);
>> dseq = arithdeco(code,counts,length(seq));
>> length(code)
477
```

dobili smo lošiji rezultat za 7 simbola, čime je ujedno pokazano kako ovakva greška utiče na tačnost kodnog postupka. Za samostalni rad pokušajte realizaciju aritmetičkih kodova iz Poglavlja III.7, a koji su složeniji nego ovi koji su pokriveni MATLAB-ovim skupom funkcija.

POGLAVLJE IV

KOMUNIKACIONI KANAL

IV. KOMUNIKACIONI KANAL

Komunikacioni kanal predstavlja fizički spojni put između predajnika i prijemnika informacije. Kanali mogu biti bežični (atmosfera, tečnosti, svemir), kablovi (optički, bakarni), memorije na računaru (hard, USB, CD, DVD i drugi diskovi), papir (za pisani tekst), usta−vazduh−uši (za govor i, na primjer, igru gluvih telefona), slike (bilo analogne bilo digitalne) itd. U kanalu se dešavaju smetnje koje otežavaju komunikaciju. Termin **smetnja** je u inženjerskoj praksi najčešće zamijenjen terminom **šum**, koji implicira neodređenost, odnosno slučajnost ove pojave. U prvom poglavlju već smo pobrojali da ove smetnje mogu biti uzrokovane prirodnim fenomenima, ljudskim aktivnostima koje uključuju namjerna ometanja i druge komunikacije. Primjer prirodnog fenomena mogu biti vremenske prilike i geografija terena, a vještačkog zgrade i drugi građevinski objekti, saobraćaj, industrijske pojave itd. Na kraju, u telekomunikacijama pojava smetnji zavisi od postupaka (modulacija) i uređaja koji se primjenjuju. Očigledno je nemoguće obuhvatiti sve ove elemente koji se mogu javiti, niti mi imamo znanja, na ovom nivou, da obuhvatimo, na primjer, uticaj fizičkih pojava na naše komunikacije. Stoga se, umjesto da se predmetne pojave detaljno analiziraju (što gotovo po pravilu nije ni moguće), pristupa modelovanju smetnje statističkim alatima, odnosno (uslovnim) vjerovatnoćama. Ovo modelovanje će biti obrađeno u prvom dijelu ovoga poglavlja. Predmetni model predstavlja **kanal bez memorije** kao najjednostavniji, ali ujedno i dalje veoma generalan i izazovan model kanala. Pojam **kapaciteta kanala** obrađen je u narednom dijelu ovog poglavlja sa fundamentalnim matematičkim definicijama i dokazima, kao i izvođenjem kapaciteta kanala za neke popularne modele kanala. Nakon toga se upuštamo u **II Šenonovu teoremu** ili **teoremu o kodiranju kanala**. Naglasak je na njenom tumačenju premda su data i detaljna matematička izvođenja, namijenjena prije svega naprednijim studentima. Četvrti dio ovog poglavlja odnosi se na neke napredne koncepte vezane za kanal, koji su ukratko pobrojani, a nisu ni namijenjeni studentima osnovnih studija. Peti dio poglavlja razmatra napredno modelovanje kanala. I ovaj dio je, prije svega, namijenjen naprednijim studentima.

IV.1 Model kanala

Komunikacioni kanal bez memorije (ako drugačije nije naglašeno, biće razmatran ovaj model kanala) modeluje se statističkim modelom medija preko kojeg prolaze signali nosioci informacija. Sve fizičke pojave, odabrane modulacione tehnike i komunikacioni uređaji apstrahuju se skupom uslovnih vjerovatnoća da se simboli iz predajnog (ulaznog) alfabeta: $A=\{a_1,a_2,\ldots,a_N\}$ transformišu u simbole prijemnog (izlaznog) alfabeta $B=\{b_1,b_2,\ldots,b_M\}$. Označimo vjerovatnoću da se simbol s ulaza a_i pojavio na izlazu kao simbol b_j, kao: $P(b_j \mid a_i)$, gdje je $i \in [1,N]$ i $j \in [1,M]$. Veličina dva alfabeta A i B ne mora biti ista, ali o ovome ćemo nešto više reći u nastavku. Radi pojednostavljenja, uslovne vjerovatnoće često zapisujemo kao $P(b_j \mid a_i)=P_{i,j}$. Uslovne vjerovatnoće se često upisuju u matricu koja se naziva **matrica tranzicije**. U literaturi se često mijenja redosljed vrsta i kolona ove matrice, ali smo u ovom udžbeniku usvojili konvenciju da se ulazni simboli upisuju u redove, a izlazni u vrste matrice:

$$\mathbf{P}=\begin{bmatrix} P(b_1 \mid a_1) & P(b_2 \mid a_1) & \cdots & P(b_M \mid a_1) \\ P(b_1 \mid a_2) & P(b_2 \mid a_2) & \cdots & P(b_M \mid a_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(b_1 \mid a_N) & P(b_2 \mid a_N) & \cdots & P(b_M \mid a_N) \end{bmatrix}=\begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,M} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N,1} & P_{N,2} & \cdots & P_{N,M} \end{bmatrix}.$$

Matrica tranzicije stanja ima sljedeće osobine: *i*-ti red odgovora *i*-tom ulaznom simbolu a_i; *j*-ta kolona odgovara *j*-tom izlaznom simbolu b_j. Suma elemenata u redu uvijek je jednaka 1 (ovo se grubo može protumačiti kao – ako neki simbol uđe u kanal, sigurno će nešto izaći iz njega):

$$\sum_{j=1}^{M} P_{i,j}=\sum_{j=1}^{M} P(b_j \mid a_i)=1.$$

Ako je $p(a_i)$ vjerovatnoća simbola a_i, važi:

$$\sum_{i=1}^{N}\sum_{j=1}^{M} P(b_j \mid a_i)p(a_i)=1.$$

Vjerovatnoće simbola a_i i b_j se odnose kao:

$$\begin{aligned} &p(a_1)P_{1,1}+p(a_2)P_{2,1}+\ldots+p(a_q)P_{q,1}=p(b_1) \\ &p(a_1)P_{1,2}+p(a_2)P_{2,2}+\ldots+p(a_q)P_{q,2}=p(b_2) \\ &\ldots \\ &p(a_1)P_{1,s}+p(a_2)P_{2,s}+\ldots+p(a_q)P_{q,s}=p(b_s) \end{aligned}$$

ili u simplifikovanoj notaciji:

$$\sum_{i=1}^{N} p(a_i)P(b_j \mid a_i) = p(b_j) \quad j = 1,2,...,q,$$

odnosno, u matričnoj formi, vektori vjerovatnoća simbola ulaznog $p(\mathbf{a}) = [p(a_1), p(a_2),\ldots,p(a_N)]^T$ i izlaznog alfabeta $p(\mathbf{b}) = [p(b_1), p(b_2),\ldots,p(b_M)]^T$ odnose se kao:

$$p(\mathbf{b}) = \mathbf{P}^T p(\mathbf{a}).$$

Da bismo izbjegli suvoparno opisivanje matematičkih definicija vezanih za model komunikacionog kanala, uvedimo jedan od najjednostavnijih, a opet izuzetno bitan i primjenljiv – model **binarnog simetričnog kanala** (u praksi često nazivan prosto BSC, engl. *binary symmetric channel*), koji je prikazan na Slici IV.1. Pretpostavka je da se simboli ulaznog alfabeta {0,1} prenose ispravno sa vjerovatnoćom Q, a da do greške dolazi sa vjerovatnoćom greške P (vjerovatnoća greške po bitu se često označava **ber**, od engl. *bit error rate*).

Tranziciona matrica ovog kanala je stoga simetrična (odakle i naziv kanala):

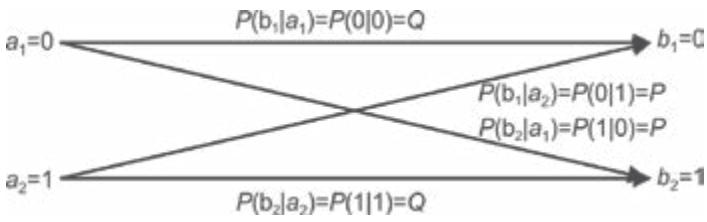
$$\mathbf{P} = \begin{bmatrix} Q & P \\ P & Q \end{bmatrix}$$

jer smo usvojili da je $P(0 \mid 0) = P(1 \mid 1) = Q$ i $P(1 \mid 0) = P(0 \mid 1) = P$. Neka su vjerovatnoće ulaznih simbola $p(a = 0) = p$ i $p(a = 1) = 1 - p$, pa se može uspostaviti sljedeća jednakost kanala:

$$pQ + (1-p)P = p(b = 0)$$
$$pP + (1-p)Q = p(b = 1). \square$$

Entropije i međusobna informacija su osnovni alati koji se koriste za određivanje performansi kanala. Podsjetimo se fundamentalnih veza između ovih veličina iz Poglavlja II:

$$H(A,B) = H(B) + H(A \mid B) = H(A) + H(B \mid A)$$
$$I(A;B) = H(A) + H(B) - H(A,B) = H(A) - H(A \mid B) = H(B) - H(B \mid A) \geq 0.$$



Slika IV.1. Grafički prikaz binarnog simetričnog kanala

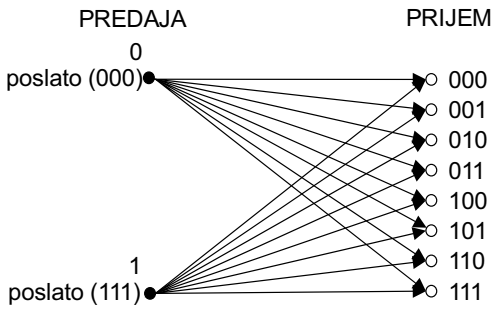
Fundamentalne granice za ove veličine su:

$$0 \le H(A \mid B) \le H(A) \text{ i } 0 \le H(B \mid A) \le H(A) \text{ i } H(A,B) \le H(A) + H(B),$$

sa jednakošću koja važi samo ako su događaji u ova dva alfabeta međusobno nezavisni.

U primjeru binarnog simetričnog kanala imali smo da su predajni alfabet A i prijemni alfabet B s istim elementima, odnosno s istim brojem simbola. U praksi, međutim, to ne mora biti slučaj. Česta je situacija da je broj simbola prijemnog alfabeta veći od broja simbola predajnog. Veoma rijetko se dešava obrnuta situacija da je broj simbola predajnog veći od onog kod prijemnog. Razlog za ovo je u greškama koje se mogu desiti u kanalu. Po pravilu, mi u kanal šaljemo digitalne poruke {0,1} predstavljene nekom fizičkom veličinom, npr. naponom od 0 i 5V. Zbog smetnji u kanalu, dolazi do distorzije poslatog signala, pa na prijemu mogu biti naponski nivoi u kontinualnom domenu. Na prijemu postoji kolo koje vrši diskretizaciju i uobličavanje primljenih signala kako bi se vratili na digitalni oblik. Istina, kao što će biti pokazano u Poglavljima VII i VIII, danas postoje uspješni kodovi kanala koji dekodiranje obavljaju s vrijednostima iz kontinualnog domena. Drugi razlog za pojavu većeg broja simbola prijemnog alfabeta, od onoga kod predajnog, jeste u kodiranju kanala. Kodiranje kanala dodaje redundanciju u kanal kako bi se eventualne greške koje nastaju u kanalu mogle ispraviti. Na ovaj način, broj poruka koje se mogu primiti na prijemu znatno je veći od broja poslatih poruka.

Posmatrajmo slučaj kada želimo putem kanala za prenos informacija prenijeti simbole {0,1}. U saznanju smo da se u kanalu mogu dogoditi greške (neka je vjerovatnoća greške po bitu P i neka su greške međusobno nezavisne). Primitivan način kodiranja kanala je da, umjesto jednog simbola, šaljemo triput isti simbol, čime omogućujemo dekodiranje na osnovu **većinske** – **majoritetne logike** (kod ima više naziva, kao što je npr. **repetitivni – ponavljajući kod**). Dekodiranje se obavlja većinom glasova. Ako su primljene dvije ili tri jedinice, donosi se odluka da je poslati simbol jedinica, a u suprotnom da je poslati simbol nula. Dakle, u



Slika IV.2. Grafički prikaz prenosa informacija za repetitivni kod

ovom slučaju u kanal šaljemo dvije moguće sekvence {000, 111}, a mogući su prijemi svih kombinacija od tri bita {000, 001, 010, 011, 100, 101, 110, 111}, odnosno više imamo mogućih primljenih nego poslatih poruka; dok je suprotna situacija rijetkost, ova je uobičajena. Ovaj sistem je grafički prikazan na Slici IV.2, dok mu je matrica tranzicije:

$$\begin{array}{l} 0\rightarrow \\ 1\rightarrow \end{array}\begin{bmatrix} (1-P)^3 & (1-P)^2P & (1-P)^2P & (1-P)P^2 & (1-P)^2P & (1-P)P^2 & (1-P)P^2 & P^3 \\ P^3 & (1-P)P^2 & (1-P)P^2 & (1-P)^2P & (1-P)P^2 & (1-P)^2P & (1-P)^2P & (1-P)^3 \end{bmatrix}$$

$$\begin{array}{cccccccc} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111. \end{array}$$

IV.2 Kapacitet kanala

Kapacitet kanala je mjera koja kaže koliko se informacija može dobiti korišćenjem kanala po nekoj jedinici korišćenja (simbolu poslate informacije ulaznog alfabeta). Od fundamentalne je važnosti za razumijevanje svih procesa u kanalu koji utiču na dizajn predajnika i prijemnika (odnosno kodera i dekodera kanala).

Definicija IV.1: Kapacitet kanala se definiše kao maksimalna međusobna informacija: $C = \max_{p(x)} I(A;B)$, gdje je maksimum tražen preko svih mogućih distribucija simbola ulaznog alfabeta (ovdje označeno u argumentu maksimuma kao $p(x)$). □

Način da se ovo razumije je putem primjera. Primjer će biti najjednostavniji mogući, a opet veoma praktični binarni simetrični kanal.

Primjer IV.1. Određivanje kapaciteta binarnog simetričnog kanala. Međusobna informacija se može definisati na više načina, a ovdje upotrijebimo sljedeću vezu međusobne informacije i entropija:

$$I(A;B) = H(B) - H(B \mid A) = H(B) - \sum_{i=1}^{2} p(a = a_i) H(B \mid a = a_i).$$

Prvo izvršimo izvođenje po definiciji. Vjerovatnoća da je B jednako 0, odnosno 1 može se izraziti kao:

$$p(b=0) = p(1-P) + (1-p)P$$
$$p(b=1) = pP + (1-p)(1-P).$$

Dalje, u slučaju da znamo da je poslato 0, događaj B uzima vrijednost $b=0$ sa vjerovatnoćom $(1-P)$, a vrijednost $b=1$ sa vjerovatnoćom P. Stoga slijedi da je entropija:

$$H(B \mid a=0) = -P\log P - (1-P)\log(1-P) = H(P).$$

Na isti način:

$$H(B \mid a=1) = H(P).$$

Dakle, drugi sabirak u izrazu za međusobnu informaciju je:

$$H(P)p + H(P)(1-p) = H(P),$$

odnosno, determinisan je samo vjerovatnoćom greške u kanalu P, to jest fizičkim uslovima u kanalu koje smo apstrahovali ovom veličinom.

Sada idemo direktno, težim putem da odredimo kapacitet kanala:

$$\begin{aligned} I(A;B) = &-[p(1-P)+(1-p)P]\log[p(1-P)+(1-p)P] \\ &-[pP+(1-p)(1-P)]\log[pP+(1-p)(1-P)] - H(P). \end{aligned}$$

Vidimo da međusobna informacija zavisi samo od p (vjerovatnoće simbola izlaznog alfabeta), dok su parametri kanala za dati kanal nepromjenljivi, odnosno dati za posmatrane okolnosti. Da bismo odredili kapacitet kanala, moramo maksimizovati međusobnu informaciju, što se u našem slučaju svodi na maksimizaciju po p. Dakle, sada bi trebalo da bude jasno šta znači formulacija: „preko svih mogućih distribucija ulaznog alfabeta“. Traži se raspodjela simbola ulaznog alfabeta koja će dati maksimum međusobne informacije. Slično kao i u vodovodnim cijevima, maksimum je vrijednost najvećeg kapaciteta, odnosno maksimalne isporuke vode (u našem slučaju informacije) koja je za dati (vodovodni) kanal moguća.

Odredimo izvor međusobne informacije po p:

$$\begin{aligned} \frac{\partial I(A;B)}{\partial p} &= -[1-2P]\log[p(1-P)+(1-p)P] - \frac{[p(1-P)+(1-p)P]}{[p(1-P)+(1-p)P]}[1-2P]\log e \\ &-[2P-1]\log[pP+(1-p)(1-P)] - \frac{[pP+(1-p)(1-P)]}{[pP+(1-p)(1-P)]}[2P-1]\log e = 0 \\ &= -[1-2P]\log[p(1-P)+(1-p)P] + [1-2P]\log[pP+(1-p)(1-P)] = \\ &= [1-2P]\log\frac{pP+(1-p)(1-P)}{p(1-P)+(1-p)P} = 0. \end{aligned}$$

Odavde slijedi da je

$$\begin{aligned} &\frac{pP+(1-p)(1-P)}{p(1-P)+(1-p)P} = 1 \\ &pP+(1-p)(1-P) = p(1-P)+(1-p)P \\ &p(2P-1)+(1-p)(1-2P) = 0 \\ &(2P-1)(2p-1) = 0 \end{aligned}$$

što za svako P daje rješenje $p = 1/2$. Uvrštavanjem ove vrijednosti u izraz za međusobnu informaciju, slijedi:

$$C = \max_{p} I(A;B) = 1 - H(P). \square$$

Prije nego protumačimo ovaj veoma bitan rezultat, vrijedi reći da smo mogli da izbjegnemo računanje izvoda koristivši jednostavne osobine entropije. Naime, kada smo izveli relaciju:

$$I(A;B) = H(B) - H(P),$$

mogli smo da odmah zapišemo konačnu relaciju za kapacitet jer se entropija binarnog alfabeta maksimizuje na 1 bit.

Tumačimo sada formulu za kapacitet binarnog simetričnog kanala. Pretpostavimo da u kanal šaljemo 1 bit informacije. Na izlazu, zbog grešaka u kanalu, možemo da dobijemo maksimalno $1 - H(P)$ bita. Dakle, greške koje se dešavaju u kanalu možemo da shvatimo kao rupu iz koje, prilikom distribucije informacije (vode), dio informacija curi i ne stiže na odredište. Stoga, ako želimo da na prijemu imamo 1 bit korisne informacije, u kanal moramo da dodamo određenu redundanciju. Za kapacitet se ponekad kaže da se mjeri u **bitima po korišćenju** (odnosno koliko korisne informacije dobijamo na prijemu po svakom simbolu koji pošaljemo). Radi jednostavnosti, mi ćemo kapacitet mjeriti prosto u bitima.

Kapacitet kanala se nalazi u sljedećim granicama:

$$0 \le C \le \min[\log \| A \|, \log \| B \|],$$

gdje je $\| \|$ broj elemenata u ulaznom, odnosno izlaznom alfabetu. Ove granice su posljedica osobina međusobne informacije i entropija: međusobna informacija je nenegativna veličina, te:

$$I(A;B) = H(A) - H(A \mid B) \le H(A) \le \log \| A \|$$

$$I(A;B) = H(B) - H(B \mid A) \le H(B) \le \log \| B \|.$$

Za svaki dati kanal može se odrediti kapacitet, odnosno maksimum međusobne informacije. Može se pokazati da je međusobna informacija kontinualna funkcija raspodjele simbola ulaznog alfabeta $p(x)$, te da je konkavna (ispupčena) funkcija od $p(x)$, što znači da posjeduje maksimum. Nema potrebe da ovo dokazujemo, ali možemo vidjeti u prethodnom primjeru da kada smo izrazili $I(A;B)$ u funkciji p, dobili smo funkciju kojoj smo bez većih problema odredili maksimum.

Minimalan kapacitet binarnog simetričnog kanala postiže se za $P = 1/2$ i iznosi $C = 0$. Dakle, ako imamo kanal sa 50% grešaka, na njegovom kraju ne možemo da dobijemo nikakvu informaciju. Za $P = 1$ dobijamo da je kapacitet jednak $C = 1$

bit (Slika IV.3) jer se kompletna korisna informacija može dobiti iz primljene poruke prostom negacijom. Stoga se, obično, vjerovatnoće greške veće od $P = 1/2$ ne uzimaju u razmatranje, već se prosto smatra da se ovakve situacije mogu svesti na situaciju za $P < 1/2$ invertovanjem primljene poruke.

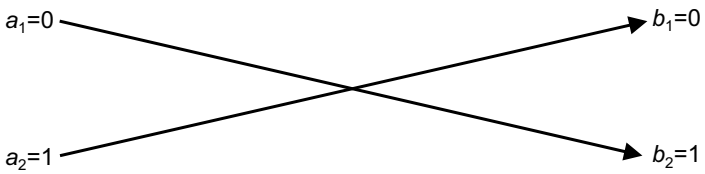
Primjer IV.2. Posmatrajmo binarni simetrični kanal sa brisanjem i bez greške. Predmetni kanal je prikazan na Slici IV.4. Generičko **stanje brisanja** označeno je sa *e* (engl. *erasure*). Podrazumijeva situaciju kada prijemnik, zbog velikog slabljenja signala u kanalu, nije u stanju da detektuje da je nešto uopšte primljeno. Pretpostavili smo da je sistem simetričan, te da se brisanje dešava s istom vjerovatnoćom α, bez obzira koji od dva ulazna simbola je poslan.

Za određivanje kapaciteta kanala sa brisanjem, pođimo od istog izraza za međusobnu informaciju kao u prethodnom slučaju:

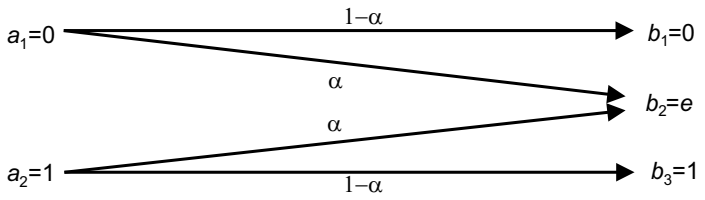
$$I(X;Y) = H(Y) - H(Y \mid X) = H(Y) - H(\alpha).$$

Ovdje smo uočili da je $H(Y \mid X) = H(\alpha)$, jer ako je na ulazu u sistem 0, moguća stanja na izlazu su 0 i *e*, koja se pojavljuju sa vjerovatnoćama $\{1-\alpha, a\}$. Dakle, ovo se može tretirati kao binarni događaj s entropijom $H(\alpha)$, odnosno $H(Y|X=0) = H(\alpha)$. Na isti način zaključujemo da je $H(Y|X=1) = H(\alpha)$. Odavde slijedi da je:

$$H(Y \mid X) = \sum_{i=1}^{N} p(x_i) H(Y \mid X = x_i) = p(X=0)H(Y \mid X=0) + p(X=1)H(Y \mid X=1) =$$
$$= p(X=0)H(\alpha) + p(X=1)H(\alpha) = H(\alpha)[p(X=0) + p(X=1)] = H(\alpha).$$



Slika IV.3. Slučaj binarnog simetričnog kanala sa $P = 1$



Slika IV.4. Simetrični kanal sa brisanjem i bez greške

Ostaje da se odredi kolika je entropija $H(Y)$, a za to je potrebno da odredimo kolike su vjerovatnoće pojavljivanja simbola alfabeta Y. Ovdje je bitno napomenuti da ne možemo postići da je maksimalno $H(Y) = \log 3$ jer je ograničenje kapaciteta vezano i za broj simbola ulaznog i broj simbola izlaznog alfabeta za bilo koju distribuciju ulaznog alfabeta, što će biti pokazano. Radi jednostavnosti označavanja, uzmimo da je $P(X=1)=p$:

$$P(Y=0)=P(X=0)(1-\alpha)=(1-p)(1-\alpha)\ P(Y=e)=P(X=0)\alpha+P(X=1)\alpha=\text{a}$$

$$P(Y=1)=P(X=1)(1-\alpha)=p(1-\alpha).$$

Odavde slijedi:

$$\begin{aligned}H(Y) &= -[(1-p)(1-\alpha)\log(1-\alpha)(1-p)+\alpha\log\alpha+p(1-\alpha)\log p(1-\alpha)] = \\ &= -(1-p)(1-\alpha)\log(1-p)-(1-p)(1-\alpha)\log(1-\alpha)-\alpha\log\alpha- \\ &\quad -p(1-\alpha)\log p-p(1-\alpha)\log(1-\alpha) \\ &= (1-\alpha)[-p\log p-(1-p)\log(1-p)] \\ &\quad -(1-\alpha)\log(1-\alpha)[(1-p)+p]-\alpha\log\alpha = \\ &= (1-\alpha)H(p)-(1-\alpha)\log(1-\alpha)-\alpha\log\alpha = \\ &= (1-\alpha)H(p)+H(\alpha).\end{aligned}$$

Dakle, međusobna informacija kod ovoga sistema je jednaka:

$$I(X;Y)=H(Y)-H(\alpha)=(1-\alpha)H(p)+H(\alpha)-H(\alpha)=(1-\alpha)H(p).$$

Maksimum međusobne informacije, računat preko svih mogućih raspodjela simbola ulaznog alfabeta, dobija se za $H(p)$ maksimalno, odnosno za $H(p)=1$ bit. Dakle, kapacitet kanala sa brisanjem, a bez greške iznosi:

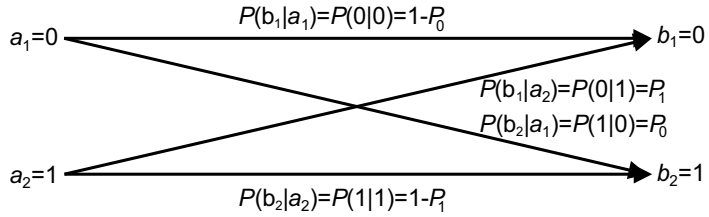
$$C=1-\alpha.\ \square$$

Možemo zaključiti da kod ovog tipa kanala brisanje za svaki poslati bit informacije „krade“ α bita informacije koji ne stiže na prijemnu stranu.

Kanal sa brisanjem, a bez greške može se opisati putem matrice tranzicije, kao:

$$\begin{bmatrix}1-\alpha & \alpha & 0\\ 0 & \alpha & 1-\alpha\end{bmatrix}.$$

U oba prethodna slučaja imamo kanale kod kojih postoji određena simetrija (kasnije će pojam simetrije kanala biti uveden mnogo preciznije). Međutim, određivanje kapaciteta kanala mnogo je složenije za slučaj kada su kanali bez simetrije. Posmatrajmo sljedeći primjer, koji je na prvi pogled prilično sličan kao prethodna dva.



Slika IV.5. Model binarnog nesimetričnog kanala

Primjer IV.3. Odrediti kapacitet binarnog nesimetričnog kanala. Pretpostavimo da je greška prilikom prenosa nule jednaka P_0, dok je greška prilikom prenosa simbola jedan jednaka P_1 i da ove dvije veličine mogu biti različite $P_0 \neq P_1$. Kanal je prikazan na Slici IV.5.

Rješenje: Riječ je o prilično složenom problemu iako na prvi pogled izgleda jako jednostavan, tek nešto složeniji od onog kod binarnog simetričnog kanala. Situacija, nažalost, nije takva. Označimo sa p vjerovatnoću slanja simbola 0 sa prijemnika. Započnimo s određivanjem kapaciteta na osnovu istog izraza za međusobnu informaciju koji smo prethodno koristili:

$$I(X;Y) = H(Y) - H(Y|X).$$

Događaj Y je binaran s entropijom $H(p(1-P_0)+(1-p)P_1)$. Uslovna entropija $H(Y|X)$ je jednaka:

$$H(Y \mid X) = pH(P_0) + (1-p)H(P_1).$$

Međusobna informacija se sada može zapisati kao:

$$I(X;Y) = f(p;P_0,P_1) = H(p(1-P_0)+(1-p)P_1) - pH(P_0) - (1-p)H(P_1).$$

Da bismo odredili kapacitet kanala, potrebno je da maksimizujemo međusobnu informaciju diferencirajući predmetni izraz po p:

$$\frac{\partial f(p;P_0,P_2)}{\partial p} = (1-P_0-P_1)\log_2 \frac{1-[p(1-P_0)+(1-p)P]}{p(1-P_0)+(1-p)P_1} - H(P_0) + H(P_1) = 0$$

$$2^{[H(P_0)-H(P_1)]/(1-P_0-P_1)} = \frac{1-[p(1-P_0)+(1-p)P]}{p(1-P_0)+(1-p)P_1}$$

$$p(1-P_0)+(1-p)P_1 = \frac{1}{2^{[H(P_0)-H(P_1)]/(1-P_0-P_1)}+1}$$

$$p(1-P_0-P_1) = \frac{1}{2^{[H(P_0)-H(P_1)]/(1-P_0-P_1)}+1} - P_1$$

$$p = \frac{1}{(1-P_0-P_1)}\frac{1}{2^{[H(P_0)-H(P_1)]/(1-P_0-P_1)}+1} - \frac{P_1}{(1-P_0-P_1)}.$$

Uvrštavanjem ovog izraza u relaciju za međusobnu informaciju, dobijamo izraz za kapacitet:

$$C = \frac{1}{1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}}\log_2\left[1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}\right] - \frac{2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}}{1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}}\log_2\left[\frac{2^{2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}}}{1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}}\right]$$

$$-\frac{H(P_0)}{1-P_0-P_1}\frac{1}{1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}} - \frac{P_1H(P_0)}{1-P_0-P_1} -$$

$$-H(P_1)\left[1-\frac{1}{1-P_0-P_1}\frac{1}{1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}}+\frac{1}{1-P_0-P_1}\right]$$

$$=\log_2\left[1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}\right] - 2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}\frac{H(P_0)-H(P_1)}{1-P_0-P_1} - \frac{H(P_0)-H(P_1)}{1-P_0-P_1}\frac{1}{1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}}$$

$$-\frac{P_1H(P_0)+H(P_1)(1-P_0)}{1-P_0-P_1} =$$

$$=\log_2\left[1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}\right] - \frac{H(P_0)-H(P_1)-P_1H(P_0)-H(P_1)(1-P_0)}{1-P_0-P_1} =$$

$$=\log_2\left[1+2^{\frac{H(P_0)-H(P_1)}{1-P_0-P_1}}\right] - \frac{H(P_0)(1-P_1)-H(P_1)P_0}{1-P_0-P_1}.$$

Dakle, nesimetričnost unosi veliku složenost u proces određivanja kapaciteta kanala. Ta složenost često dovodi do toga da u zatvorenom obliku nije moguće izvesti izraz za kapacitet, pa se nerijetko prilazi simulacijama da bi se kapacitet odredio. Iz prethodno izvedene formule slijedi da za slučaj simetričnog kanala sa $P_0 = P_1 = P$ imamo:

$$C = \log_2(1) - H(P)(1-2P)/(1-2P) = 1 - H(P),$$

što se očekivano svodi na slučaj binarnog simetričnog kanala. □

Nakon što smo vidjeli teorijski značaj simetričnih kanala, u stanju smo da pređemo na formalno definisanje simetričnosti kanala.

Definicija IV.2. Kanal se naziva **strogosimetričnim** ako su svi redovi tranzicione matrice (matrice uslovnih vjerovatnića ulaza/izlaza kanala) permutacije (isti elementi, ali raspoređeni na razne načine) i sve kolone tranzicione matrice su permutacije. Često se strogosimetričan kanal naziva simetričnim. Kanal se naziva **sla-**

bosimetričnim ako su redovi (vrste) tranzicione matrice permutacije, dok su sume kolona matrice jednake. Pod kanalom **simetričnim po blokovima** podrazumijeva se onaj koji ima kolone koje se mogu grupisati u strogosimetrične podmatrice. □

Očigledno je binarni simetrični kanal strogosimetričan. Pretpostavimo da imamo kanal sa K-ulaznih simbola i isto toliko izlaznih kod kojeg se vjerovatnoća greške P uniformno raspoređuje preko svih simbola. Tranziciona matrica ovog kanala je:

$$\mathbf{P}=\begin{bmatrix} 1-P & P/(K-1) & \cdots & P/(K-1) \\ P/(K-1) & 1-P & \cdots & P/(K-1) \\ \vdots & \vdots & \ddots & \vdots \\ P/(K-1) & P/(K-1) & \cdots & 1-P \end{bmatrix}.$$

Ovaj kanal ispunjava uslov stroge simetričnosti. Stroga simetričnost slično važi i kod sljedećeg primjera ternarnog kanala:

$$\mathbf{P}=\begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.1 & 0.7 \\ 0.1 & 0.7 & 0.2 \end{bmatrix}.$$

Posmatrajmo sada slučaj kanala koji ima dva simbola ulaznog alfabeta i četiri simbola izlaznog alfabeta sa tranzicionom matricom:

$$\mathbf{P}=\begin{bmatrix} 1/3 & 1/4 & 1/4 & 1/6 \\ 1/6 & 1/4 & 1/4 & 1/3 \end{bmatrix}.$$

Ovaj kanal je slabosimetričan jer su elementi u oba reda isti, dok je suma vjerovatnoća po kolonama jednaka konstanti 1/2.

Kanal sa brisanjem, a bez greške ne pripada ni simetričnim ni kanalima koji su simetrični u širem smislu, već se matrica tranzicije može dekomponovati na dvije matrice koje zadovoljavaju simetričnost:

$$\mathbf{P}=\begin{bmatrix} 1-\alpha & \alpha & 0 \\ 0 & \alpha & 1-\alpha \end{bmatrix} \Rightarrow \mathbf{P}'=\begin{bmatrix} 1-\alpha & 0 \\ 0 & 1-\alpha \end{bmatrix} \quad \mathbf{P}''=\begin{bmatrix} \alpha \\ \alpha \end{bmatrix}.$$

Svaki simetrični kanal je istovremeno slabosimetričan. Sljedeća teorema je od krupnog značaja za pojednostavljivanje računanja kapaciteta kanala, a važi i za simetrične kanale kao specijalni slučaj slabosimetričnih kanala.

Teorema IV.1. Kapacitet slabosimetričnih kanala je jednak

$$C=\log||Y||-H(\mathbf{r})=\log M-H(\mathbf{r}),$$

gdje je $||Y||=M$ kardinalnost (broj simbola izlaznog alfabeta), dok je $H(\mathbf{r})$ entropija redova tranzicione matrice.

Dokaz. Međusobna informacija ovog kanala je jednaka:

$$I(X;Y) = H(Y) - H(Y \mid X) = H(Y) - H(\mathbf{r}).$$

Naime, uslovna entropija $H(Y|X)$ može se sračunati kao:

$$H(Y \mid X) = \sum_{i=1}^{N} p(x_i) H(Y \mid X = x_i).$$

Ovdje uslovne entropije $H(Y \mid X = x_i)$ predstavljaju entropije vrsta tranzicione matrice, a kako su elementi ovih vrsta uvijek jednaki (sa različitim permutacijama elemenata), zaključujemo da je $H(Y \mid X = x_i) = H(\mathbf{r})$, pa slijedi da je:

$$H(Y \mid X) = H(\mathbf{r}) \sum_{i=1}^{N} p(x_i) = H(\mathbf{r}).$$

Maksimum međusobne informacije (kapacitet kanala) dobija se onda maksimizacijom entropije $H(Y)$, koja postiže maksimum od $\log||Y|| = \log M$, čime smo dokazali predmetnu teoremu. □

Teorema IV.1 veoma pojednostavljuje računanje kapaciteta kanala u mnogim praktičnim slučajevima.

Primjer IV.4. Posmatrajmo ternarni kanal sa tranzicionom matricom:

$$\mathbf{P} = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.1 & 0.7 \\ 0.1 & 0.7 & 0.2 \end{bmatrix}.$$

Odrediti kapacitet ovoga kanala.

Rješenje: Na osnovu teoreme o kapacitetu slabosimetričnog kanala, slijedi:

$$C = \log 3 - H(\mathbf{r}) = \log 3 + 0.7 \log 0.7 + 0.2 \log 0.2 + 0.1 \log 0.1 = 0.4281 \text{ bit.} \square$$

Kapacitet kanala koji je simetričan po blokovima određuje se na nešto složeniji način. Vidjeli smo da smo razdvojili tranzicionu matricu na dvije podmatrice, od kojih prva predstavlja matricu binarnog simetričnog kanala. Međutim, ako se posmatra odvojeno od ostatka, ovaj dio kanala je kanal bez greške (ovo je i potrebno da bi matrica tranzicije **P'** bila ispravka sa sumama pojedinih vrsta koje su jednake 1). Kapacitet ovoga dijela kanala ja $C' = 1$ bit. Drugi dio kanala (predstavljen matricom **P "**) predstavlja brisanje kada se nula i jedinica pretvaraju u stanje brisanja. Kapacitet ovoga dijela kanala je $C'' = 0$ bita, jer smo sigurni u ishod, pa se na izlasku iz kanala ne dobija nikakva informacija, bez obzira na ono što u kanal ulazi. Kapacitet ovog kanala je onda:

$$C = \sum_{l=1}^{s} p(l) C^{(l)},$$

gdje su $p(l)$ vjerovatnoće da se kanal nalazi u nekom od podblokova (podstanja), a $C^{(l)}$ su kapaciteti tih podstanja, gdje je s ukupan broj ovakvih simetričnih podmatrica. U našem slučaju imamo dva podstanja: stanje brisanja, u koje ulazimo s vjerovatnoćom α i stanje kada nema brisanja, u koje ulazimo s vjerovatnoćom $1-\alpha$. Tako je kapacitet kanala sa brisanjem, a bez greške:

$$C = C'(1-\alpha) + C''\alpha = 1-\alpha.$$

U praksi se dijeljenje matrice u simetrične minore pod blokovimaa nešto rjeđe koristi nego teorema za slabosimetrične kanale jer je složenije za primjenu, te je ponekad nije moguće ni primijeniti direktno.

Kapacitet kanala je direktno povezan s našim zahtjevima za resurse u komunikacionom sistemu, na primjer sa **propusnim opsegom** koji je potreban za prenos informacija. Obično imamo na raspolaganju neki fiksni frekvencijski opseg za prenos podataka B. Broj simbola (odbiraka) signala koji se mogu prenijeti, odnosno broj bita, proprocionalan je sa propusnim opsegom cB (c je konstanta proporcionalnosti za nas manje bitna). Preko navedenog kanala mi možemo prenijeti cBC (za svaki odbirak signala mi prenosimo količinu informacija C). Predmetna veličina se mjeri u bit/s (ako je kapacitet izražen u bitima). Ponekad se koristi i veličina koja se naziva **iskorišćenost kanala** G/cBC, koja govori koliki se stvarni dio kapaciteta kanala koristi u datoj komunikaciji.

Detalji o fizičkim karakteristikama kanala prevazilaze domen interesovanja ovog udžbenika.

IV.3 Druga Šenonova teorema

II Šenonova teorema često se naziva i **teoremom o kodiranju kanala**. Istorijski, vjerovatno, predstavlja najznačajniji rezultat u ovoj oblasti. Uspostavlja vezu između fizičkih karakteristika kanala, predstavljenih preko kapaciteta kanala, sa karakteristikama koda koji se primjenjuje za prenos informacija preko posmatranog kanala (predstavljen preko kodnog odnosa datog koda).

Definicija IV.3. Kodni odnos predstavlja odnos logaritma broja ispravnih kodnih riječi koda sa dužinom kodne riječi:

$$R = \frac{\log_2 M}{n}. \ \square$$

Na primjer, kod ASCII koda imamo $n = 8$ bita za kodiranje riječi, od kojih je 7 korisnih (informacionih). Moguće je generisati $M = 2^7 = 128$ različitih ispavnih kodnih riječi. Stoga je u ovom slučaju kodni odnos $R = 7/8$. Smisao ove veličine je da kaže koliko bita korisne informacije (u prosjeku) dobijamo za dati kod na osnovu jednog bita u kodnoj riječi. Dakle, zaključujemo da je kodni odnos veoma

jasna fizička veličina za posmatrani kod. Cilj kodiranja kanala je da se u postupku dekodiranja dobije informacija koja je poslata bez greške. Označimo sa $\lambda^{(n)}$ vjerovatnoću greške u procesu dekodiranja za neki kod sa kodnom riječju dužine n (smatramo da bez gubitka opštosti sve kodne riječi u procesu kodiranja kanala imaju istu dužinu).

Definicija IV.4. Kodni odnos R naziva se dostižnim ako za $n \to \infty$ postoji kod koji daje $\lambda^{(n)} \to 0$. □

Česta nedoumica postoji vezano za kodni odnos kada $n \to \infty$. Suštinski, kako dužina kodne riječi raste tako raste i broj ispravnih kodnih riječi, tako da je moguće držati da je limes kodnog odnosa R konstanta koja je različita od nule i beskonačnosti.

Teorema IV.2. II Šenonova teorema (ponekad se naziva Šenon−Hartlijevom teoremom). Svi kodni odnosi $R \leq C$ su dostižni. □

Verbalno može da se kaže da je za dati kanal (kanal sa datim kapacitetom C) moguće konstruisati kod sa kodnim odnosom koji je manji ili jednak kapacitetu kanala, a koji daje vjerovatnoću greške, u procesu dekodiranja, koja teži nuli za kodnu riječ čija dužina teži beskonačno.

Suprotnost kodne teoreme je da za $R > C$ nije moguće konstruisati kod koji daje proizvoljno malu vjerovatnoću greške.

U literaturi se može naći veliki broj dokaza ove teoreme za opšti slučaj i mnoštvo specijalnih slučajeva. Svi ovi dokazi su izuzetno komplikovani i malo (ako išta) govore o načinu kako se kodovi mogu konstruisati. Stoga, sve što slijedi do kraja dokaza kodne teoreme stavljeno je pod zvjezdice i ostavljeno za one kojima su interesantna teorijska razmatranja, te za polaznike napredne verzije našeg kursa.

*IV.3.1 Definicije i teoreme potrebne za dokaz kodne teoreme**

Definicija IV.5. Skup $A_\varepsilon^{(n)}$ združenih tipičnih sekvenci $\mathbf{x}^n = (x_1, x_2, ..., x_n)$ i $\mathbf{y}^n = (y_1, y_2, ..., y_n)$, s združenom raspodjelom $p(x,y)$, zadovoljava:

$$A_\varepsilon^{(n)} = \left\{ (\mathbf{x}^n, \mathbf{y}^n) \in \mathrm{X}^n \times \mathrm{Y}^n \mid \left| -\frac{1}{n} \log p(\mathbf{x}^n) - H(X) \right| < \varepsilon \wedge \left| -\frac{1}{n} \log p(\mathbf{y}^n) - H(Y) \right| < \varepsilon \wedge \right.$$
$$\left. \left| -\frac{1}{n} \log p(\mathbf{x}^n, \mathbf{y}^n) - H(X,Y) \right| < \varepsilon \right\}.$$

U pitanju je, kao što vidimo, nešto eksplicitnija definicija tipične sekvence, u ovom slučaju generalizovana za dvije sekvence, od kojih sekvenca $\mathbf{x}^n = (x_1, x_2, ..., x_n)$

može predstavljati niz simbola koje šaljemo u kanal, a sekvenca $\mathbf{y}^n = (y_1, y_2, ..., y_n)$ niz simbola koje smo primili na prijemniku.

Definicija IV.6. Formalna definicija kodiranja kanala. Pretpostavimo da preko kanala želimo poslati neku od *M* poruka. Za svaku od poruka kreiramo kodnu riječ odgovarajuće dužine $\mathbf{x}^n(j),\ j \in [1, M]$. Na prijemniku je primljena riječ $\mathbf{y}^n(j),\ j \in [1, M]$ u kojoj postoje eventualne greške. Na osnovu odgovarajuće dekodirajuće funkcije *g*(), dekoder pokušava da iz primljene sekvence $\mathbf{y}^n(j)$ odredi *j*.

Prethodnom definicijom formulisan je kanal bez memorije. Kod kanala **sa memorijom** postoji zavisnost (definisana preko uslovnih vjerovatnoća) primljene poruke od poslate u datom i prethodnim trenucima, odnosno od prethodno dekodiranih poruka.

Greška u prenosu se može formulisati kao:

$$g(\mathbf{y}^n(j)) \neq j,$$

odnosno situacija kada dekodirajuća funkcija zbog grešaka u poruci nije u stanju da pravilno dekodira ispravnu poruku. Vjerovatnoću greške da smo poslali poruku *j*, a da se na osnovu primljene sekvence $\mathbf{y}^n(j)$ ona dekodira neispravno, označavamo kao:

$$\lambda_j = \Pr\left(g(\mathbf{y}^n(j)) \neq j\right).$$

Često se, umjesto s pojedinačnom vjerovatnoćom greške, koristi maksimalna vjerovatnoća greške računata preko svih kodnih riječi datog koda:

$$\lambda_{\max} = \max_{j \in [1, M]} \lambda_j,$$

gdje $\max_{j \in [1, M]}$ označava maksimum argumenta računat za sve vrijednosti indeksa.

Teorema IV.3. Združena asimptotska ekviparticiona osobina.

(1) Ukupna vjerovatnoća združene tipične sekvence može se učiniti proizvoljno velikom (bliskom 1), $\Pr\left\{A_\varepsilon^{(n)}\right\} > 1 - \varepsilon$ za *n* dovoljno veliko, odnosno $n \to \infty$.

(2) Broj elemenata u združenom tipičnom skupu nalazi se u granicama: $(1-\varepsilon)2^{n(H(X,Y)+\varepsilon)} \leq \left\|A_\varepsilon^{(n)}\right\| \leq 2^{n(H(X,Y)+\varepsilon)}$.

(3) Vjerovatnoća da združeni niz pripada tipičnom skupu nalazi se u granicama

$$(1-\varepsilon)2^{-n(I(X;Y)+3\varepsilon)} \leq \Pr\left\{(\mathbf{x}^n, \mathbf{y}^n) \in A_\varepsilon^{(n)}\right\} \leq 2^{-n(I(X;Y)-3\varepsilon)}.$$

Dokaz. U prethodnom poglavlju, kod opisa, vidjeli smo da logaritam združene vjerovatnoće konvergira po vjerovatnoći entropiji:

$$-\frac{1}{n}\log p(\mathbf{x}^n) \to H(X).$$

Konvergencija po vjerovatnoći znači da se za dovoljno veliko n vjerovatnoća razlike $-\frac{1}{n}\log p(\mathbf{x}^n)$ s entropijom može učiniti proizvoljno malom, odnosno pretpostavljamo da postoji neko n_1 za koje važi da je za $n > n_1$:

$$\Pr\left[\left|-\frac{1}{n}\log p(\mathbf{x}^n) - H(X)\right| > \varepsilon\right] < \frac{\varepsilon}{3}.$$

Slično se može odrediti neko n_2, gdje za $n > n_2$ važi:

$$\Pr\left[\left|-\frac{1}{n}\log p(\mathbf{y}^n) - H(Y)\right| > \varepsilon\right] < \frac{\varepsilon}{3},$$

te da se za združenu entropiju može naći $n_{3,}$ za koje važi da je za $n > n_3$:

$$\Pr\left[\left|-\frac{1}{n}\log p(\mathbf{x}^n, \mathbf{y}^n) - H(X,Y)\right| > \varepsilon\right] < \frac{\varepsilon}{3}.$$

Dakle, za $n > \max(n_1, n_2, n_3)$ vjerovatnoća unije tri skupa sigurno je manja od ε, odnosno vjerovatnoća pripadanja tipičnom skupu sigurno je veća od $1 - \varepsilon$.

Drugi dio dokaza slijedi iz činjenice da je vjerovatnoća članova tipične sekvence $2^{-n(H(X,Y)+\varepsilon)}$, te da ukupna vjerovatnoća tipične sekvence mora biti manja od 1, pa stoga broj elemenata sekvence mora biti manji od:

$$\left\|A_\varepsilon^{(n)}\right\| \le \frac{1}{2^{-n(H(X,Y)+\varepsilon)}} = 2^{n(H(X,Y)+\varepsilon)}.$$

Donja granica slijedi iz uslova konvergencije po vjerovatnoći.

Treći stav teoreme slijedi iz činjenica koje se odnose na broj elemenata u tipičnom skupu, te pod pretpostavkom o nezavisnosti događaja primijenimo asimptotsku ekviparticionu osobinu na pojedinačne sekvence $\mathbf{x}^n$ i $\mathbf{y}^n$:

$$\begin{aligned}\Pr\left\{(\mathbf{x}^n, \mathbf{y}^n) \in A_\varepsilon^{(n)}\right\} &\le 2^{n(H(X,Y)+\varepsilon)} 2^{-n(H(X)-\varepsilon)} 2^{-n(H(Y)-\varepsilon)} \\ &= 2^{n(H(X,Y)-H(X)-H(Y)-3\varepsilon)} = 2^{n(I(X;Y)-3\varepsilon)}.\end{aligned}$$

Posljednja relacija slijedi iz definicije međusobne informacije.

*IV.3.2 Dokaz druge Šenonove teoreme**

Dokaz II Šenonove teoreme. Osnovni element u ovom dokazu i jedan od ključnih Šenonovih doprinosa je slučajno generisanje kodnih riječi. Generišimo slučajno 2^{nR} kodnih riječi dužine n u skladu s distribucijom simbola ulaznog alfabeta:

$$p(\mathbf{x}^n) = \prod_{i=1}^{n} p(x_i).$$

Dobijene (slučajne) kodne riječi mogu se smjestiti u redove matrice:

$$\mathsf{C} = \begin{bmatrix} x_1(1) & x_2(1) & \cdots & x_n(1) \\ x_1(2) & x_2(2) & \cdots & x_n(2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(2^{nR}) & x_2(2^{nR}) & \cdots & x_n(2^{nR}) \end{bmatrix}.$$

Svaki element ove matrice generisan je nezavisno, u skladu s vjerovatnoćama pojedinih simbola. Dakle, vjerovatnoća generisanja partikularnog koda je:

$$\Pr(\mathsf{C}) = \prod_{w=1}^{2^{nR}} \prod_{i=1}^{n} p(x_i(w)).$$

Pretpostavka je da se komunikacija obavlja na sljedeći način.

- Slučajni kod C se generiše na opisani način. Kod je poznat i prijemniku i predajniku. Prijemnik i predajnik znaju i tranzicionu matricu uslovnih vjerovatnoća u kanalu.
- Predajnik bira neku poruku W na slučajan način, u skladu s uniformnom vjerovatnoćom (pretpostavljamo da je bilo koja poruka poslata s istom vjerovatnoćom, a napominjemo da je poruka 2^{nR}):

$$P(W = w) = 2^{-nR} \qquad w = 1, 2, ..., 2^{nR}.$$

Neka je poslata kodna riječ w kodirana sa $X^n(w)$ (predstavlja w-ti red matrice C).

- Prijemnik prima riječ Y^n u skladu sa distribucijom:

$$P(y^n \mid x^n(w)) = \prod_{i=1}^{n} p(y_i \mid x_i(w)).$$

Množenje uslovnih vjerovatnoća je posljedica pretpostavke da je u pitanju kanal bez memorije, odnosno da su uzastopni simboli, koji su primljeni na izlazu iz kanala, bez međusobnog uticaja.

- Primjemnik, na osnovu primljene sekvence, pokušava da odredi koja je poruka poslana. Ovdje usvajamo sljedeći skup pravila u dekodirajućoj proceduri, kojom

se Y^n pokušava odrediti W (odnosno dobiti procjena $\hat{W}$). Prvo provjeravamo da li postoji $X^n(\hat{W})$ tako da je $(X^n(\hat{W}), Y^n)$ tipična združena sekvenca. Takođe, potrebno je da nema više od jedne združene tipične sekvence za koju ovo važi, odnosno da ne postoji $k \neq \hat{W}$ za koje je $(X^n(k), Y^n)$ združena tipična sekvenca (u suprotnom imamo grešku).

– Greška u dekodiranju se dešava i kada je dekodirana poruka različita od poslate $W \neq \hat{W}$.

Naredni korak u Šenonovom dokazu je određivanje prosječne vjerovatnoće greške preko svih kodnih riječi u kodnoj knjizi i preko svih kodnih knjiga C. Cilj ovakvog izvođenja je da ako se prosječna vjerovatnoća greške, računata preko slučajnih kodnih knjiga, može učiniti proizvoljno malom, onda barem jedna kodna knjiga postoji sa proizvoljno malom vjerovatnoćom greške. Ovo je veoma jednostavan mehanizam kojim je Šenon pokazao postojanje dobrog koda (ali ne i kako do njega da dođemo). Uprosječena vjerovatnoća greške po svim kodnim knjigama je:

$$P_E = \sum_{\mathsf{C}} P(\mathsf{C}) P_E^{(n)}(\mathsf{C}) = \sum_{\mathsf{C}} P(\mathsf{C}) \frac{1}{2^{nR}} \sum_{w=1}^{2^{nR}} \lambda_w(\mathsf{C}) = \frac{1}{2^{nR}} \sum_{\mathsf{C}} P(\mathsf{C}) \sum_{w=1}^{2^{nR}} \lambda_w(\mathsf{C}).$$

U prethodnom izrazu $P(\mathsf{C})$ je vjerovatnoća izbora partikularne kodne knjige, dok je $P_E^{(n)}(\mathsf{C})$ vjerovatnoća da se u kodnoj riječi dužine n, iz date kodne knjige C, dogodi greška. Indeksom w označili smo riječ koja je poslata, a kako prosječna vjerovatnoća greške ne zavisi od konkretne riječi koja je poslata, možemo da pretpostavimo da je poslata ona koju indeksiramo sa $w = 1$.

Definišimo sljedeći događaj:

$$E_i = \{(\mathbf{x}^n(i), \mathbf{y}^n) \in A_\varepsilon^{(n)}\} \text{ za } i \in \{1, 2, ..., 2^{nR}\},$$

koji predstavlja situaciju da je $\mathbf{y}^n$ združeno tipična sekvenca sa i-tom kodnom riječju. Greška se javlja ako se:

– E_1^c dogodi, odnosno kada $\mathbf{y}^n$ nije združeno tipična sekvenca s prvom kodnom riječju;

– $E_2 \cup E_3 \cup \cdots \cup E_{2^{nR}}$ pojavi, odnosno kada je primljena sekvenca združeno tipična sa pogrešnom kodnom riječju.

Neka P_E označava $P_E(w = 1)$, tako da imamo:

$$P_E = P(E_1^c \cup E_2 \cup E_3 \cup \cdots \cup E_{2^{nR}}) \leq P(E_1^c) + \sum_{i=2}^{2^{nR}} P(E_i).$$

Predmetna relacija slijedi na osnovu činjenice da je vjerovatnoća unije događaja manja ili jednaka sumi vjerovatnoća pojedinačnih događaja, te da se jednakost postiže samo u slučaju kada su događaji nezavisni. Na osnovu asimptotske ekviparticione osobine, slijedi da važi:

$$P(E_1^c) \leq \varepsilon$$

za dovoljno veliko n. Ujedno, vjerovatnoća da su $\mathbf{x}^n(i)$ i $\mathbf{y}^n$ združeno tipični manja je ili jednaka $2^{-n(I(X;Y)-3\varepsilon)}$, što daje:

$$P_E \leq \varepsilon + \sum_{i=2}^{2^{nR}} 2^{-n(I(X;Y)-3\varepsilon)} = \varepsilon + (2^{nR}-1)2^{-n(I(X;Y)-3\varepsilon)} \leq \varepsilon + 2^{-n(I(X;Y)-3\varepsilon-R)} \leq 2\varepsilon,$$

za n koje je dovoljno veliko i $R < I(X;Y) - 3\varepsilon$. Dakle, ako je $R < I(X;Y)$, možemo da izaberemo ε i n, tako da je prosječna vjerovatnoća greške (prosječna po kodnim knjigama i kodnim riječima) manja od 2ε. Preostaje nam da pojasnimo kako se vrši postupak uprosječavanja po kodnim riječima i kodnim knjigama.

- Možemo usvojiti da je vjerovatnoća simbola ulaznog alfabeta takva da međusobna informacija daje kapacitet, odnosno možemo da usvojimo $R < C$.
- Pošto je pokazano da je prosječna vjerovatnoća greške, uprosječena preko svih kodnih knjiga, mala, to znači da postoji barem jedna kodna knjiga C^* sa malom prosječnom vjerovatnoćom pogreške. Napomenimo da teorema ne uspostavlja način na koji se do ove kodne knjige (odnosno kodnog postupka) može doći.
- Uočimo da najbolja polovina kodnih riječi mora da produkuje vjerovatnoću greške koja je manja od 4ε jer u suprotnom ne bismo bili u mogućnosti da dostignemo srednju vrijednost od 2ε. Gora polovina kodnih riječi može se odbaciti, čime se dobija kodni odnos:

$$R' = R - \frac{1}{n},$$

što je asimptotski zanemarivo manje u odnosu na R (za n koje teži beskonačno, praktično je isto). Stoga se može zaključiti da je maksimalna vjerovatnoća pogreške za najbolju kodnu knjigu manja ili jednaka 4ε.

Na ovaj način smo zaključili dokaz II Šenonove teoreme (o kodiranju kanala). Teorema ima brojne ekstenzije. U narednoj sekciji ćemo obraditi formulaciju teoreme za slučaj kanala koji je zahvaćen Gausovim šumom. Postoje modifikacije teoreme koje govore o dostižnoj vjerovatnoći greške u procesu dekodiranja za fiksnu dužinu kodne riječi (koja ne mora težiti u beskonačno). Brzo po uvođenju ove teoreme došlo je do dinamičnog razvoja kodova za kodiranje kanala. Nažalost, taj razvoj traje do današnjih dana i nije bio tako efektivan kao što je bio razvoj kodova za

kodiranje izvora. Tek nakon gotovo 70 godina od uvođenja ove teoreme, razvijeni su neki od kodnih postupaka koji se približavaju performansama, u smislu vjerovatnoće greške onome što teorema predviđa da je dostižno.

Potrebno je naglasiti da u literaturi postoje i drugi alternativni dokazi ove važne teoreme.

*IV.3.3 Gausovski kanal**

Česta pretpostavka je da se naši signali prenose kroz kanal u kojem se smetnje mogu modelovati kao slučajni Gausov proces (proces sa Gausovom funkcijom gustine raspodjele). Nažalost, praksa je pokazala da kanali često odstupaju od ovoga modela, ali zbog teorijske važnosti ovdje dajemo osnovna izvođenja vezana za ovaj bitan model kanala.

Posmatrajmo informaciju datu kao x_i, te da je primljena informacija na prijemniku:

$$y_i = x_i + \nu_i,$$

gdje je ν_i bijeli Gausov šum sa srednjom vrijednošću 0 i varijansom σ^2, koji se može opisati funkcijom raspodjele:

$$p(\xi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\xi^2}{2\sigma^2} \right).$$

Predmetni model se naziva aditivni jer dolazi do sabiranja signala nosioca informacije sa šumom. Gausov šum se često naziva šumom sa **normalnom raspodjelom**, koja se u predmetnom slučaju označava $\mathbf{N}(0,\sigma^2)$. Pretpostavimo da predajnik šalje nulu i jedinicu kao bipolarne impulse (vrijednost sa pozitivnim i negativnim predznakom) sa amplitudom $\sqrt{P}$ i $-\sqrt{P}$ (kvadrat amplitude predstavlja snagu signala P). Zbog šuma u kanalu, na prijemnoj strani dobijamo kontinualne vrijednosti. Odluku o primljenom bitu donosićemo poređenjem primljenog signala sa pragom. Ovdje je prirodno da se prag postavi na 0. Tada se greška dešava u situaciji kada pošaljemo $x_i = \sqrt{P}$, a primimo $y_i < 0$, te kada šaljemo $x_i = -\sqrt{P}$, a primimo pozitivan signal $y_i > 0$. Pod pretpostavkom da su biti poslati s jednakim vjerovatnoćama, slijedi da je vjerovatnoća pogreške:

$$\begin{aligned} P_E &= \frac{1}{2} P(y_i < 0 \mid x_i = \sqrt{P}) + \frac{1}{2} P(y_i > 0 \mid x_i = -\sqrt{P}) = \\ &= \frac{1}{2} P(\nu_i < -\sqrt{P} \mid x_i = \sqrt{P}) + \frac{1}{2} P(\nu_i > \sqrt{P} \mid x_i = -\sqrt{P}) = \\ &= P(\nu_i > \sqrt{P}) = 1 - \Phi(\sqrt{P} / \sigma) \end{aligned}$$

gdje je

$$Q(x)=\frac{1}{\sqrt{2\pi}}\int_x^{\infty}e^{-x^2/2}dx \qquad \Phi(x)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}e^{-x^2/2}dx.$$

Suštinski, ako je $x_i=\sqrt{P}$, greška se dešava kada šum obori ovaj signal ispod praga ($\nu_i<-\sqrt{P}$), a slično je za negativni impuls. Stoga, vjerovatnoća greške zavisi od međusobnog odnosa amplitude signala i standardne devijacije šuma.

Teorema IV.4. Kapacitet Gausovog kanala je:

$$C=\max I(X;Y)=\frac{1}{2}\log\left(1+\frac{P}{\sigma^2}\right).$$

Dokaz. Međusobna informacija je za dati kanal data kao:

$$I(x;y)=H(y)-H(y|x)=H(y)-H(x+\nu|x)=H(y)-H(\nu|x)=H(y)-H(\nu).$$

Primljeni signal je jednak sumi poslatog signala i Gausovog šuma. Odluka, međutim, zavisi isključivo od veličine šuma, pa stoji $H(x+\nu|x)=H(\nu|x)$. Pošto smatramo da je aditivni šum nezavisan od signala, dobijamo da je $H(\nu|x)=H(\nu)$. Kako je Gausov šum analogna veličina, moramo ga analizirati preko entropijske funkcije prilagođene takvim veličinama. Entropija analognih veličina (naziva se diferencijalna entropija), ponekad se označava malim slovom $h(x)$, računa se na sljedeći način (radi pojednostavljenja, uzet je prirodni logaritam):

$$\begin{aligned}H(\nu_i)&=-\int_{-\infty}^{\infty}p(\xi)\ln p(\xi)d\xi=\\&=-\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\xi^2/2\sigma^2\right)\ln\left[\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\xi^2/2\sigma^2\right)\right]d\xi=\\&=\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\xi^2/2\sigma^2\right)\ln\left[\sqrt{2\pi}\sigma\exp\left(\xi^2/2\sigma^2\right)\right]d\xi.\end{aligned}$$

Uvedimo smjenu promjenljivih $t=x\sqrt{2}$:

$$\begin{aligned}H(\nu_i)&=\frac{\sqrt{2}\sigma}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}\exp(-t^2)\ln\left[\sigma\sqrt{2\pi}\exp(t^2)\right]dt=\\&=\frac{1}{\sqrt{\pi}}\ln\sqrt{2\pi\sigma^2}\int_{-\infty}^{\infty}\exp(-t^2)dt+\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}t^2\exp(-t^2)dt\\&=\frac{1}{2}\ln 2\pi\sigma^2+\frac{1}{2}=\frac{1}{2}\ln(2\pi e\sigma^2).\end{aligned}$$

U izvođenju su, pored uobičajenih operacija vezanih za logaritamsku funkciju, korišćeni identiteti:

$$\int_{-\infty}^{\infty} \exp(-t^2)dt = \sqrt{\pi} \qquad \int_{-\infty}^{\infty} t^2 \exp(-t^2)dt = \frac{\sqrt{\pi}}{2}.$$

Maksimizacija međusobne informacije se dobija kada je maksimalna vrijednost entropije $H(y)$. Slučajna promjenljiva y takođe prati Gausovu raspodjelu (predstavlja zbir Gausove slučajne promjenljive sa korisnom porukom). Dakle, $H(y)$ se takođe može zapisati kao:

$$H(y) = \frac{1}{2}\ln(2\pi e\sigma_y^2)$$

gdje je σ_y^2 varijansa Gausovog procesa y. Najveća moguća vrijednost ove varijanse je:

$$\sigma_y^2 = P + \sigma^2,$$

odnosno kada je signaliziranje isto Gausov proces sa snagom P i potpuno nezavisan od Gausovog šuma. Dakle, kapacitet kanala je jednak:

$$C = \frac{1}{2}\ln\left(2\pi e(P+\sigma^2)\right) - \frac{1}{2}\ln\left(2\pi e\sigma^2\right) = \frac{1}{2}\ln\left(\frac{P+\sigma^2}{\sigma^2}\right) = \frac{1}{2}\ln\left(1+\frac{P}{\sigma^2}\right).$$

Sada možemo da pređemo na kapacitet izražen u bitima kao:

$$C = \frac{1}{2}\log_2\left(1+\frac{P}{\sigma^2}\right) \text{ bita}.$$

Relacija je potpuno očekivana: što je odnos snage signala sa varijansom šuma veći to je i kapacitet kanala veći, odnosno možemo preciznije da prenosimo informacije kroz kanal (ili veću količinu informacija) sa manjom potrebom za kodiranjem i redundancijom. Što je odnos signal–šum (P / σ^2) manji, to je i kapacitet kanala manji, jer u kanalu dolazi do većeg broja pogreški.

Vrijedi ukazati na činjenicu da predmetna relacija važi za sve tipove signala kod kojih je snaga signala ograničena na P.

*IV.3.4 Neka proširenja Šenonove teoreme**

Vrlo bitno proširenje kodne teoreme je da se može pokazati da nije moguć kod koji daje vjerovatnoću greške koja teži nuli kada je kodni odnos veći od kapaciteta: $R > C$. Ovu teoremu nećemo dokazivati.

Druga bitna posljedica kodne teoreme je tvrdnja da ako imamo kod sa vjerovatnoćom greške dekodiranja koja asimptotski opada ka nuli, predmetni kod sigurno zadovoljava kodnu teoremu. Svaki kod sa vjerovatnoćom greške koja asimptotski opada ka nuli mora da bude kod koji zadovoljava uslove kodne teoreme (zapamtimo da kodna teorema kaže da kod postoji, a ne koji je to konkretni kod). Pretpostavimo da nema greške u kanalu. Putem kodne teoreme imamo kod koji može da generiše 2^{nR} kodnih riječi. Uzimajući da je entropija izbora bilo koje od ovih kodnih riječi uniformna, tada je entropija jednaka $\log_2 2^{nR} = nR$. Ovu entropiju (označimo je $H(W)$) možemo povezati s entropijom događaja primljene sekvencije Y^n i međusobnom informacijom, kao:

$$nR = H(W) = H(W \mid Y^n) + I(W; Y^n).$$

Dalje mora da važi:

$$H(W) \leq H(W \mid Y^n) + I(X^n; Y^n)$$

jer je X^n funkcija tačne kodne riječi W. Po Fanoovoj relaciji slijedi:

$$H(W \mid Y^n) + I(X^n; Y^n) \leq 1 + P_e^{(n)} nR + I(X^n; Y^n).$$

Konačno, na osnovu činjenice da se međusobna informacija maksimizuje za kapacitet kanala, te da ovdje posmatramo sekvencu od n simbola, imamo da je

$$nR \leq 1 + P_e^{(n)} nR + I(X^n; Y^n) \leq 1 + P_e^{(n)} nR + nC.$$

Vjerovatnoća greške ograničena je sa

$$P_e^{(n)} \geq 1 - \frac{1}{nR} - \frac{C}{R}.$$

Kada n teži beskonačno, jedini način da se možemo spustiti do vjerovatnoće greške koja je nula jeste da je $R < C$.

Posljednji stav koji će ovdje biti diskutovan je združeno kodiranje izvora i kodiranje kanala. Pitanje koje je mučilo naučnike u ranim danima teorije informacije i kodova je da li se kodiranje izvora mora vršiti u nekoj vezi sa kodiranjem kanala. Prosječna dužina kodne riječi je, kao što znamo, veća ili jednaka entropiji izvora. Zatim, kodni odnos mora biti manji od kapaciteta kanala. Kapacitet kanala je povezan s entropijom izvora koja utiče na međusobnu informaciju čiji je maksimum kapacitet. Kodni odnos sa druge strane je vezan za fizičku kodnu riječ, gdje koder izvora pokušava da umanji prosječnu dužinu kodne riječi. Dakle, u objema teoremama pojavljuju se fizičke veličine vezane za dužinu konkretnih kodova s entropijama vezanim za statistike kanala i vjerovatnoćama pojavljivanja pojedinih simbola. Bez potrebe za uvođenjem precizne formulacije ovog problema, odgovarajuće teoreme i dokaza, za nas je najvažniji stav da ne postoji potreba da se

kombinuje postupak kodiranja izvora i kodiranja kanala, već da se mogu obaviti nezavisno. Odatle, zapravo, potiče i model sistema za prenos informacija koji je uveden u Poglavlju I.

IV.3.5 Primjeri kodova za drugu Šenonovu teoremu

Kodovi za kodiranje kanala biće razmatrani u poglavljima koja slijede. Ovdje će biti razmatrane, bez ulaska u samu suštinu postupka kodiranja, performanse nekih od kodova za kodiranje kanala.

Primjer IV.5. Posmatrajmo kôd kod kojega imamo $n = 7$ bita u kodnoj riječi i koji je u stanju da ispravi jednu pogrešku u kodnoj riječi. Kod ima kodni odnos $R = 4/7$. Odrediti vjerovatnoću greške za slučaj vjerovatnoće greške koja zadovoljava uslove kodne teoreme za binarni simetrični kanal.

Rješenje: Binarni simetrični kanal ima kapacitet:

$$C = 1 - H(p).$$

Kritična granica za kodnu teoremu je $C = R$, odnosno $C = 1 - 4/7 = 3/7$. Može da se pokaže da se ova kritična granica postiže za $p \approx 0.087$ ($H(0.087) \approx 4/7$). Predmetni kod radi korektno kada su svi biti koji se prenose preneseni korektno, te kada se dogodi jedna pogreška koju kod može da koriguje. Greška se dešava u ostalim slučajevima:

$$\mathrm{P}_E = 1 - (1-p)^7 - 7p(1-p)^6 = 1 - (1-p)^6(1+6p).$$

Za kritičnu granicu od $p \approx 0.087$ dobijamo:

$$\mathrm{P}_E \approx 0.1185.$$

Dobijena je prilično velika vjerovatnoća greške da se ovakva situacija dogodi. Dakle, neoptimalnost kodnog postupka, uz činjenicu da smo izvršili kodiranje malom dužinom kodne riječi, daje veliku vjerovatnoću greške, koja bitno odstupa od onoga što nam predviđa kodna teorema za veoma dugačke kodne riječi.

Primjer IV.6. Posmatrajmo dva primjera kodova koji imaju kodni odnos $R = 1/3$. Prvi kod ima $n = 3$ i u stanju je da ispravi jedan bit u kodnoj riječi. Drugi kod ima $n = 15$ i u stanju je da ispravi 3 bita u kodnoj riječi. Provjerite i grafički prikažite vjerovatnoće greške na kritičnoj vjerovatnoći greške koja korespondira s kapacitetom kanala kod kojega je $R = C$.

Rješenje: Ponovo, kapacitet kanala je $C = 1 - H(p)$, a kritična vjerovatnoća na kojoj se „sastaju“ kodni odnos i kapacitet je ona kod koje je $H(p) = 1 - C = 2/3$. Vjerovatnoća greške za koju se dostiže predmetna entropija je $p \approx 0.174$. Vjerovatnoća greške kod prvog koda (da se desi više od jedne pogreške) je:

$$P_E' = 1-(1-p)^3 - 3p(1-p)^2 \approx 0.0803.$$

Kod drugoga koda, to je vjerovatnoća da se desi više od 3 pogreške:

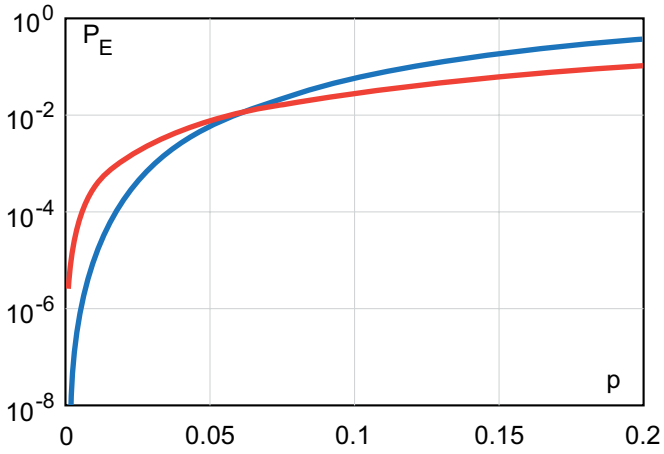
$$P_E'' = 1-(1-p)^{15} - 15p(1-p)^{14} - 105p^2(1-p)^{13} - 455p^3(1-p)^{12} \approx 0.2569.$$

Ponovo je dobijen „neočekivani“ rezultat. Ne samo da oba slučaja daju vjerovatnoće greške koje su mnogo veće od nule već i drugi slučaj, za koji po II Šenonovoj teoremi očekujemo da će biti bliži nuli, daje veću vjerovatnoću greške.

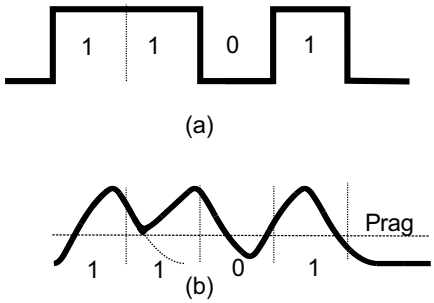
Na Slici IV.6 prikazane su vjerovatnoće greške P_E' i P_E'' za vjerovatnoće greške po bitu $p < 0.2$. Da bi dijagrami bili jasniji y-osa je data u logaritamskoj skali. Vidimo da drugi kod daje manju vjerovatnoću greške (plava linija) od prvoga (crvena linija) za $p < 0.06$. Ovo je, zapravo, i zona nekog normalnog rada kodnih sistema i u praksi je obično jedino što je od interesa. Na primjer, za $p = 0.01$ ova razlika je više od 20 puta u korist drugog koda, dok je za $p = 0.03$ to više od tri puta. Dakle, da sublimiramo, uslovi kodne teoreme nisu zadovoljeni za kodove sa kraćim kodnim riječima, ali se pravilnost bolje ispoljava kod malih vjerovatnoća greške po bitu, a koje su od praktičnog značaja.

IV.4 Kanal – složenija razmatranja*

Do sada smo razmatrali isključivo **kanal bez memorije**. Kod ovog najjednostavnijeg tipa kanala, jedan simbol izlaza zavisi samo od jednog simbola ulaza. Od trenutka kada se simbol alfabeta proslijedi kroz kanal, on putuje neko vrijeme, pa trenutak prijema kasni u odnosu na trenutak kada je simbol poslat. Međutim, ovdje



Slika IV.6. Vjerovatnoće ispravnog dekodiranja kod dva koda s istim kodnim odnosom za različite dužine kodne riječi (plava linija – kod dužine $n = 15$; crvena linija – kod dužine $n = 3$)



Slika IV.7. Ilustracija distorzija na poruci: (a) Originalna poruka; (b) Primljena digitalna poruka izobličena u kanalu

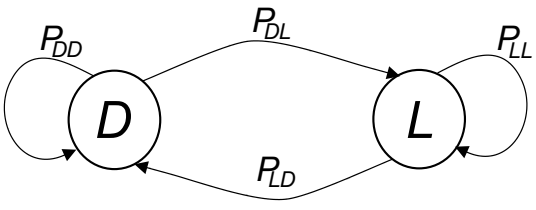
pretpostavljamo da je primljeni simbol, odnosno bit, zavisan od tačno određenog simbola/bita poslatog u kanal. Takođe, ovo podrazumijeva da je naredni simbol koji je primljen uzrokovan narednim simbolom koji je poslat u kanal. Ovo je značajna idealizacija stanja koje se dešava u kanalu, iz više razloga. Ovdje ćemo pomenuti samo tri razloga zbog kojih su kanali bez memorije prilično rijetka pojava: smetnje u kanalu nisu nezavisne od trenutka slanja, odnosno kada se dogodi neka negativna pojava vjerovatno će ona imati uticaj na veći broj poslatih simbola (npr. grmljavina ili neka ljudska djelatnost); kanali mogu da budu **disperzivni**, odnosno da putovanje jednoga simbola kroz kanal bude različitog vremena trajanja (npr. zbog kretanja učesnika u mobilnim komunikacijama) ili se može dogoditi da zbog različitih efekata (posebno, ponovo zbog kretanja prijemnika i/ili predajnika) dolazi do promjene frekvencije (Doplerovog pomjeraja) primljenog signala, te preklapanja i pomjeranja različitih simbola po vremenu; zbog nesavršenosti sistema, vremenska forma signala nosioca informacije se izobličava i prelazi preko više susjednih simbola. Demonstrirajmo primjer posljednje devijacije. U fizičkoj realizaciji sistema za prenos informacija biti su obično prikazani kao vremenske funkcije određenog trajanja, na primjer četvrtke (Slika IV.7). Zbog nelinearnosti, frekvencijske ograničenosti kanala, filtriranja i drugih problema, ove četvrtke su često izobličene i obuhvataju i susjedne signalizacione intervale. Na Slici IV.7 prikazan je slučaj sekvence 1101, gdje vidimo izobličenje signala usljed ovih efekata. Na strani prijema, dekoder kanala obično pravi odluku na osnovu poređenja sa pragom (ovdje je prag prikazan kao vertikalna isprekidana linija), pa se, na primjer, u dijelu trećeg bita – nule može lako dogoditi da, usljed šuma (stohastičkih pojava), dođe do greške u dekodiranju bita, kao djelimične posljedice deformacije na signalu u procesu prenosa (a ne isključivo greške na samom posmatranom bitu). Dakle, kada imamo ovakve distorzije, očigledno imamo međusobnu vezu između uzastopnih bita; odnosno, pojava greške na ulazu u prijemnik zavisi kako od stanja u kanalu tako i od toga kakva je poruka poslata. Kanali kod kojih se dešavaju ovakve pojave nazivaju se kanalima sa memorijom. U praksi, ovakvi

kanali su česti, a njihova analiza podrazumijeva rad s uslovnim vjerovatnoćama, kao i napredno fizičko modelovanje pojava u kanalu. Stoga, u ovom dijelu ćemo izbjeći dalje razmatranje ove problematike. Koga ova složena, premda važna − kako teorijski tako i praktično, problematika interesuje, dodatne informacije može da potraži u literaturi navedenoj na kraju knjige. Ovdje posebno izdvajamo nedavno publikovanu tezu (referenca [51]).

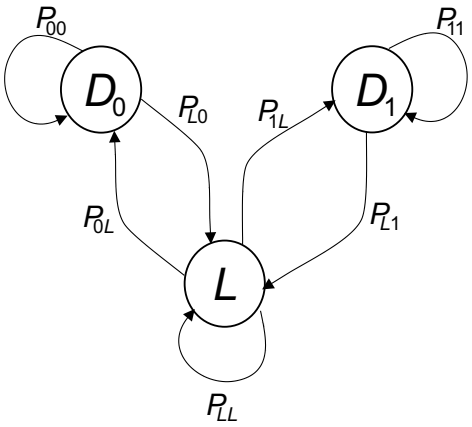
*IV.5 Drugi modeli kanala**

U praksi su razvijani brojni modeli kanala koji se koriste za slučaj kanala kod kojeg se dešavaju rijetke, ali uzastopne greške. Ovakve greške, između ostalog, izaziva rad automobilskih motora, grmljavine, namjerno ometanje itd. Stoga su se razvili modeli kanala kod kojih postoji jedna relativno mala greška u normalnim uslovima, dok je u nepovoljnim uslovima vjerovatnoća greške velika i blizu 0.5. Takav kanal je modelovao Gilbert (engl. *E. N. Gilbert*), pa se po njemu i naziva Gilbertovim (ili ponekad Gilbert−Eliotovim) modelom. Model ima dva stanja D (dobro) i L (loše). Svako stanje kanala ponaša se kao posebni binarni simetrički kanal. Pored toga, postoje vjerovatnoće za prelazak iz jednog u drugo stanje P_{DD} (iz dobrog u dobro), P_{DL} (iz dobrog u loše), P_{LD} (iz lošeg u dobro) i P_{LL} (iz lošeg u loše). Model je prikazan na Slici IV.8. Obično su vjerovatnoće da se ostane u stanju znatno veće od vjerovatnoća da se stanje promijeni. Takođe, vjerovatnoća da se napusti dobro stanje obično je mnogo manja nego da se napusti loše stanje. Tipičan primjer je: $P_{DD}=0.99$, $P_{DL}=0.01$, $P_{LD}=0.15$, $P_{LL}=0.85$. Postoji više načina kako se sada može modelovati izlaz iz kanala u ovom slučaju. Jedan način je da su dobro i loše stanje dva binarna kanala od kojih je prvi s malom vjerovatnoćom greške, a drugi je s velikom vjerovatnoćom greške. Drugi način za modelovanje je da je dobro stanje 0, a da je loše stanje 1. Rezultat koji produkuje kanal dobija se superponiranjem rezultata binarnih simetričnih kanala, tako da jedinica iz Gilbertovog modela označava grešku u prenosu. Kanali u kojima se javljaju uzastopne (engl. *burst*) greške mogu se dobiti iz modela pogodnim izborom parametara.

Naredni napredni model kanala je Smit–Boven–Džojsov model, koji je razvijen u davna vremena, kada se detaljno razmatrala problematika prenosa informacija



Slika IV.8. Gilbertov model

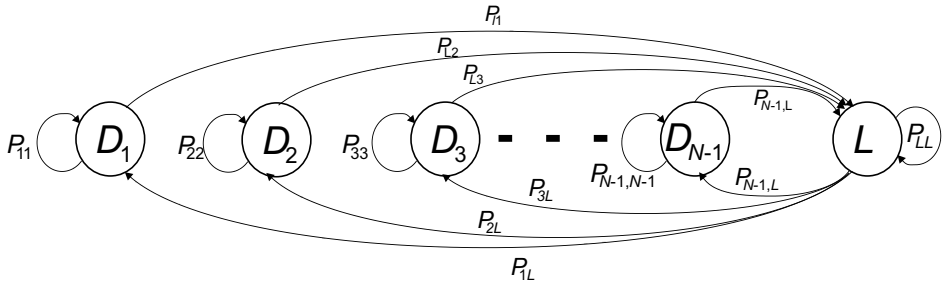


Slika IV.9. Smit–Boven–Džojsov model kanala

preko telefonskog kanala (sa bakarnim kablovima – žicama). Jasno je da danas dobrim dijelom odstupa od navedene problematike. Međutim, teorijski elementi i modeli razvijeni u to doba značajni su i danas. Ovdje dajemo samo elementarne informacije o ovom kanalnom modelu, a mnogo više podataka o njegovom razvoju i primjeni može se naći u referenci [52]. Za sve koji planiraju da rade sa modelima komunikacionog kanala, bilo u praksi ili u daljem naučnom istraživanju, preporučljivo je da u svojoj biblioteci imaju ovo davno izdanje, na sreću, dostupno danas u elektronskom obliku. Kod ovog modela postoje dva dobra stanja D_0 i D_1 i loše stanje L, povezani kao na Slici IV.9. Dobra stanja su karakterisana malim (eventualno i bez grešaka) vjerovatnoćama greške, dok su greške u lošem stanju bliske 0.5. Kao što se može vidjeti sa slike, nema konekcije između dobrih stanja. Napomenimo da smo dobra stanja D_0 i D_1 indeksirali sa 0 i 1, dok je loše stanje L indeksirano slovnom oznakom. Tako $P_{11} = P(D_1|D_1)$ i $P_{L1} = P(L|D_1)$. Matrica tranzicije se može prikazati kao:

$$\mathbf{P} = \begin{bmatrix} P_{00} & 0 & P_{0L} \\ 0 & P_{11} & P_{1L} \\ P_{L0} & P_{L1} & P_{LL} \end{bmatrix}.$$

Naredna nadogranja modela kanala je Fričmenov model (engl. *Fritchman*, ponekad se zove i Fričmen−Svobodin). Ranih 1960-ih, Fričmen je došao na ideju da kanal modeluje u više stanja, od kojih su neka stanja dobra, a neka loša. Postoji više ekstenzija ovog kanalnog modela, ali kod svake postoji određeni broj međusobno redno povezanih dobrih stanja, u čijem se nastavku nalazi određeni broj loših stanja. Pokazano je da ovakav pristup daje veoma dobro modelovanje stanja kod brojnih praktičnih situacija u radio-kanalima. Često se uzima da postoji samo jedno loše



Slika IV.10. Fričmen−Svobodin model kanala sa jednim lošim stanjem L i N-1 dobrih stanja

i $N-1$ dobro stanje. Kod ovako pojednostavljenog modela kanala (Slika IV.10) pretpostavlja se da se u svakom dobrom stanju može ostati određeno vrijeme, te da se iz dobrog stanja može preći samo u loše stanje, dok se iz lošeg stanja može preći u bilo koje dobro stanje.

Premda je ovaj model kanala dosta star, fleksibilnost, u smislu mogućnosti da se odabere veliki broj parametara (vjerovatnoća), omogućava podesivost na različite oblike savremenih, a veoma bitnih kanala i modulacionih standarda. Naravno, ovo je samo dio praktičnih modela kanala koji se danas koriste, jer se razvijaju i drugi modeli sa razvojem komunikacija, a posebno s potrebama za širokopojasne komunikacije (prije svega, rad multimedijalnih aplikacija i interneta).

Premda se u ovom udžbeniku, kako je to dosta uobičajeno, posmatraju smetnje i pojave koje se mogu modelovati kao Gausov ili uniformni šum, vrijedi istaći da su kanali koji imaju takve karakteristike relativno rijetki u praksi. Stoga ćemo ovdje ukratno pomenuti i nekoliko drugih tipova slučajnih promjenljivih i opisati u osnovi njihove karakteristike.

Uniformni šum, kao što smo rekli, modeluje se funkcijom gustine raspodjele:

$$p_u(\xi) = \frac{1}{b-a} \quad \xi \in [a,b] \text{ i } p_u(\xi) = 0 \text{ drugdje.}$$

Uglavnom ga koristimo za generisanje grešaka koje su i.i.d. procesi.

Gausov slučajni šum (često se kaže šum sa normalnom raspodjelom) modeluje se funkcijom gustine raspodjele u obliku Gausove zvonaste krive:

$$p_g(\xi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\xi-\mu)^2}{2\sigma^2} \right),$$

gdje su μ, σ i σ^2, redom: srednja vrijednost, standardna devijacija i varijansa šuma. Gausov šum je veoma popularan model smetnji i drugih pojava u kanalu. Razlozi su višestruki. Prvi je činjenica da, po poznatoj centralnoj graničnoj teoremi, više nezavisnih izvora šuma daju združeni efekat koji se može modelovati kao Gausov šum. Drugi razlog je u činjenici da su optimalni filteri i druga rješenja koja se primjenjuju kod Gausovih pojava linearni, odnosno da se mogu se efikasno realizovati. Nažalost, kanal se u komunikacijama rijetko kada može modelovati na ovaj način sa zadovoljavajućom preciznošću, jer obično imamo jedan ili mali broj izvora smetnji koji dominiraju nad ostalima. Podsjetimo se šale „da je konačno nađen kanal za naše kodove" kada su naučnici po prvi put opservirali ovakav kanal u komunikaciji sa vještačkim satelitima. Slučajni procesi koji karakterišu uslove u kojima naši kodovi funkcionišu rijetko su gausovski, a još rjeđe su i.i.d. Međutim, zbog složenosti razmatranja sistema, obično se poseže za pojednostavljenim modelima, pa se onda kodovi prilagođavaju realnim okolnostima.

Već smo pominjali, u prethodnom poglavlju, Laplasov šum, odnosno Laplasov model. Funkcija gustine raspodjele definisana je na sličan način kao kod Gausovog šuma:

$$p_l(\xi) = \frac{1}{2b}\exp\left(-\frac{|\xi-\mu|}{b}\right),$$

gdje je ponovo μ srednja vrijednost, dok je varijansa vezana za b kao $\sigma^2 = 2b^2$. Ono što Laplasov šum čini upotrebljivim je činjenica da sa većom vjerovatnoćom uzima velike vrijednosti, impulse, nego što je to slučaj kod Gausovog šuma. Kako je pojava rijetkih, ali snažnih smetnji karakteristična za brojne praktične kanale, Laplasov model je bitan.

Naredna distribucija šuma koju uvodimo je Rejlijeva (engl. *Rayleigh*), koja je definisana samo za nenegativne slučajne promjenljive funkcijom gustine raspodjele:

$$p_{Ray}(\xi) = \frac{\xi}{\sigma^2}\exp\left(-\frac{\xi^2}{2\sigma^2}\right), \quad \xi \geq 0.$$

Bitna karakteristika iz koje potiče važnost ove raspodjele jeste u tome da ako imamo signal s nezavisnim realnim i imaginarnim dijelom, koje su Gausove slučajne promjenljive s istim statistikama (srednjom vrijednošću i varijansom), amplituda ovog signala se ponaša u skladu s Rejlijevom raspodjelom. Slično se dešava i kod komunikacija kod tzv. signala u kvadraturi, kao što su oni kod kvadraturne modulacije kod kojih se signal moduliše različitim funkcijama (sinusoidalnim i kosinusoidalnim), gdje je rezultujuće ponašanje ponovo u skladu s Rejlijevom raspodjelom. Ovdje σ^2 nije varijansa šuma, već varijansa ulazne komponente Gausovog šuma. Srednja vrijednost i varijansa Rejlijevog šuma su, redom:

$\sigma\sqrt{\pi/2}$ i $\sigma^2\left[(4-\pi)/2\right]$. Kako u praksi postoje mnoge pojave, odnosno modulacije u komunikacijama koje imaju slično ponašanje, Rejlijeva raspodjela je od velikog značaja.

Rajsova distribucija (dobila ime po Stivenu Rajsu, engl. *Stephen Rice*) definisana je funkcijom gustine raspodjele:

$$p_{Rice}(\xi)=\frac{\xi}{\sigma^2}\exp\left(-\frac{(\xi^2+\nu^2)}{2\nu^2}\right)I_0\left(\frac{\xi\nu}{\sigma^2}\right),$$

gdje je $I_0(\xi)$ modifikovana Beselova funkcija prve vrste nultog reda:

$$I_0(\xi)=\sum_{m=}^{\infty}\frac{\xi^{2m}}{2^{2m}m!(m+1)!}.$$

Rajsova distribucija dobro modeluje realne pojave u radio-kanalima, posebno situaciju da signal bude oslabljen od samoga sebe. Naime, dešava se da u radio-kanalu primamo signal dobijen preko više putanja (engl. *multipath*). Neke od putanja produkuju signal sa fazom koja je suprotna od faze glavne komponente, te povremeno dolazi do poništavanja ili znatnog slabljenja signala na prijemu. Ova pojava se naziva **fedingom**, i faktor $\nu^2/2\sigma^2$ govori o prirodi i jačini fedinga. Srednja vrijednost i varijansa ovoga šuma mogu se naći u literaturi i predstavljeni su nešto složenijim izrazima u odnosu na prethodne.

Puasonova distribucija (fr. *Poisson*, po francuskom matematičaru Simonu Denisu Puasonu) model je signal-zavisnog šuma, kod kojeg je varijansa šuma proporcionalna amplitudi signala. Dakle, što je signal jači, to je i varijansa jača, i obrnuto. Premda je model jednostavan, on je u praksi složen za razmatranje i veoma kompleksan za kodiranje. Često se modeluje i putem množenja originalnog signala slučajnom promjenljivom, što se naziva **multiplikativnim šumom**. U praksi se javlja posebno pri akviziciji slike (multimedijalni podaci su od izuzetnog značaja za savremeni svijet, a to znači da se i teorija informacija i kodova moraju sa posebnom pažnjom odnositi prema ovoj oblasti).

Konačno, ovaj brzi pregled modela šuma, koji predstavlja samo djelić onoga što se javlja u teoriji i praksi, završavamo sa još jednim praktično bitnim modelom šuma. Opet ćemo ga koristiti za modelovanje **impulsa** − rijetkih, ali izuzetno snažnih šumova. Već smo rekli da se Laplasov šum koristi za ovu namjenu. Međutim, Laplasov model ipak ima određene mane. Ako pogledate funkciju raspodjele Gausovog šuma, ona se ponaša po zakonu $e^{-|\xi|}$, a za malo ξ ovo je približno $1-|\xi|$, odnosno ima „špicast“ oblik. Tačno je da se ovakve karakteristike obično uvode za modelovanje impulsnog šuma, ali nam je jednako bitno da dobro modeluju i mali šum, koji se pojavljuje najčešće, a „špicasta“ karakteristika nije realna u

praksi. Jedan od modela koji ima zaobljeniju karakteristiku oko $\xi = 0$ je Košijev (po slavnom francuskom matematičaru i akademiku Avgustinu Košiju – fr. *Augustin Cauchy*, mada se često, posebno u fizici, imenuje i po drugim imenima):

$$p_c(\xi) = \frac{1}{\gamma\pi}\frac{\gamma^2}{\xi^2 + \gamma^2}.$$

Košijev šum ima beskonačnu varijansu (što je nerealno u praksi) premda parametar γ kontroliše impulsivnost šuma i neka je vrsta pandana varijansi. Kako to obično biva, ova mana Košijeve raspodjele uticala je na razvoj drugih, boljih modela impulsnog šuma.

IV.6 Zadaci i softverska realizacija

IV.6.1 Riješeni zadaci

4.1. Odrediti kapacitet kanala koji prenosi ternarne poruke. Vjerovatnoća prenosa ispravnog simbola je P, dok je vjerovatnoća prenosa pogrešnog Q (po $Q/2$ da će biti prenijeta ostala dva simbola).

Rješenje: Tranziciona matrica u ovom slučaju je:

$$\begin{bmatrix} P & Q/2 & Q/2 \\ Q/2 & P & Q/2 \\ Q/2 & Q/2 & P \end{bmatrix}.$$

Stoga se, na osnovu teoreme za slabosimetrične kanale, lako pokazuje da je kapacitet ovog kanala:

$$C = \log 3 - H(\mathbf{r}) = \log 3 + P\log P + 2(Q/2)\log(Q/2) =$$

$$= \log 3 + P\log P + Q\log Q - Q = \log 3 - H(P) - Q = \log 3 - H(P) - (1-P)/2.$$

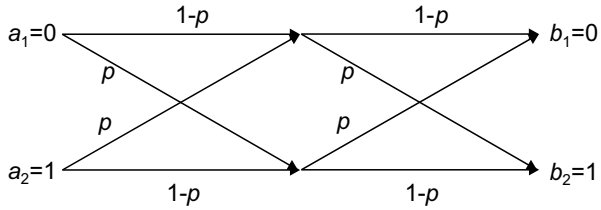
4.2. Izvedite (na osnovu relacije za slabosimetrične kanale) formulu za kapacitet opšteg simetričnog kanala kod kojeg je vjerovatnoća greške uniformno raspodijeljena na sve simbole.

Rješenje: Pretpostavimo da imamo vjerovatnoću greške P koja je uniformno raspodijeljena na sve simbole, odnosno na $N-1$ simbol koji može da predstavlja pogrešku. U tom slučaju, kapacitet kanala dat je relacijom:

$$C = \log_2 N + (N-1)P/(N-1)\log_2[P/(N-1)] + (1-P)\log_2(1-P) =$$

$$= \log_2 N + P\log_2[P/(N-1)] + (1-P)\log_2(1-P) =$$

$$= \log_2 N - P\log_2(N-1) - H(P).$$



Slika IV.11. Kaskada dva binarna simetrična kanala

4.3. Na Slici IV.11 prikazana je kaskada dva simetrična binarna kanala. Odrediti kapacitet ovoga kanala kao i kapacitet kaskade binarnih simetričnih kanala proizvoljne dužine u zavisnosti od broja kanala u kaskadi.

Rješenje: Kapacitet ovog kanala može se opisati formulom za slabosimetrične kanale, kao:

$$C = 1 - H(p_e),$$

gdje je p_e ekvivalentna vjerovatnoća greške ovog sistema. Posmatrajmo jednostavan eksperiment. Ako šaljemo 1, koja je vjerovatnoća da primimo 0? Potrebno je da napravimo grešku u prvom dijelu sistema p i da u narednoj kaskadi uspješno prenesemo simbol $(1-p)$ ili da u prvoj kaskadi napravimo tačan prenos $(1-p)$, a u drugoj pogrešan p. Dakle, ekvivalentna vjerovatnoća pogreške je:

$$p_e = 2p(1-p).$$

Stoga je kapacitet jednak:

$$C = 1 - H(2p(1-p)).$$

Na isti način se može odrediti i kapacitet kaskade n binarnih simetričnih kanala. Vjerovatnoća greške je jednaka sumi vjerovatnoća neparnog broja promjena stanja:

$$p_e = \sum_{i=1}^{\lfloor (n+1)/2 \rfloor} \binom{n}{2i-1} p^{(2i-1)} (1-p)^{n-2i+1},$$

gdje je $\lfloor \ \rfloor$ najveći cijeli broj, ne veći od argumenta.

4.4. Dokazati da je kod binarnog simetričnog kanala sa vjerovatnoćom pogreške u prenosu P_g i vjerovatnoćom brisanja simbola P_b (ovo je vjerovatnoća da na osnovu poslatog simbola nećemo moći donijeti odluku o primljenom) kapacitet jednak:

$$C = (1 - P_g - P_b) \log \left[\frac{2(1 - P_b - P_g)}{1 - P_b} \right] + P_g \log \left[\frac{2P_g}{1 - P_b} \right].$$

Da li se ovdje može koristiti relacija za slabosimetrične kanale?

Rješenje: Matrica tranzicije ovoga kanala je data kao:

$$\mathbf{P} = \begin{bmatrix} 1-P_g-P_b & P_b & P_g \\ P_g & P_b & 1-P_g-P_b \end{bmatrix}.$$

Na Slici IV.12 prikazan je predmetni model kanala. Kao što se vidi iz matrice tranzicije, ovaj kanal nije slabosimetričan u opštem slučaju jer sume vjerovatnoća po kolonama nisu jednake. Postoji jedino specijalni slučaj $P_g = P_b = 1/3$, koji ovdje nije od interesa (pogledati zadatak 4.8).

Međusobna informacija je jednaka:

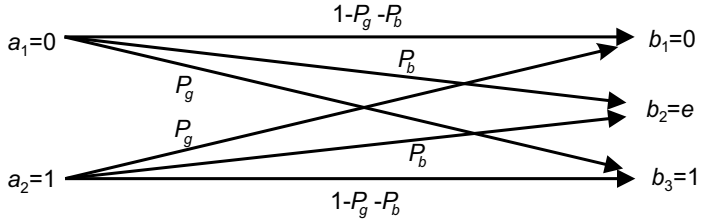
$$\begin{aligned} H(Y \mid X) &= P[-(1-P_b-P_g)\log_2(1-P_b-P_g) - P_g \log_2 P_g - P_b \log_2 P_b] \\ &\quad + (1-P)[-(1-P_b-P_g)\log_2(1-P_b-P_g) - P_g \log_2 P_g - P_b \log_2 P_b] = \\ &= -(1-P_b-P_g)\log_2(1-P_b-P_g) - P_g \log_2 P_g - P_b \log_2 P_b. \end{aligned}$$

Da bismo odredili entropiju događaja Y, moramo da znamo vjerovatnoće pojedinih simbola. Događaj 0 se dešava sa vjerovatnoćom: $P(1-P_g-P_b)+(1-P)P_g$, dok se događaj 1 dešava sa vjerovatnoćom $(1-P)(1-P_g-P_b)+PP_g$, dok je vjerovatnoća stanja brisanja P_b. Dakle, sada možemo zapisati da je entropija $H(Y)$ jednaka:

$$\begin{aligned} H(Y) = &-[P(1-P_g-P_b)+(1-P)P_g]\log_2[P(1-P_g-P_b)+(1-P)P_g] - \\ &-[(1-P)(1-P_g-P_b)+PP_g]\log_2[(1-P)(1-P_g-P_b)+PP_g] - P_b \log_2 P_b. \end{aligned}$$

Međusobna informacija je jednaka:

$$\begin{aligned} &I(X;Y) = H(Y) - H(Y|X) = \\ &-[P(1-P_g-P_b)+(1-P)P_g]\log_2[P(1-P_g-P_b)+(1-P)P_g] - \\ &-[(1-P)(1-P_g-P_b)+PP_g]\log_2[(1-P)(1-P_g-P_b)+PP_g] + (1-P_b-P_g) \\ &\log_2(1-P_b-P_g) + P_g \log_2 P_g. \end{aligned}$$



Slika IV.12. Model simetričnog kanala sa brisanjem i greškom

Međusobna informacija je očigledno funkcija od P, tako da diferenciranjem po P i izjednačavanjem sa 0, dobijamo:

$$-[(1-P_g-P_b)-P_g]\log[P(1-P_g-P_b)+(1-P)P_g]-(1-P_g-P_b-P_g)$$
$$-[-(1-P_g-P_b)+P_g]\log[(1-P)(1-P_g-P_b)+PP_g]-(-(1-P_g-P_b)+P_g)=0.$$

Poslije trivijalnih transformacija, izraz se svodi na:

$$\log[P(1-P_g-P_b)+(1-P)P_g]-\log[(1-P)(1-P_g-P_b)+PP_g]=0.$$

Ovo dalje znači da argumenti logaritma moraju biti jednaki, što konačno dovodi do toga da je jednakost zadovoljena za $P=1/2$. Dakle, dobijamo da je kapacitet jednak:

$$C=I(X;Y)=-[0.5(1-P_g-P_b)+0.5P_g]\log_2[0.5(1-P_g-P_b)+0.5P_g]-$$
$$-[0.5(1-P_g-P_b)+0.5P_g]\log_2[0.5(1-P_g-P_b)+0.5P_g]+(1-P_b-P_g)$$
$$\log_2(1-P_b-P_g)+P_g\log_2P_g$$
$$=-(1-P_b)\log_2[0.5(1-P_b)]+(1-P_b-P_g)\log_2(1-P_b-P_g)+P_g\log_2P_g.$$

Prvi član u izrazu možemo zapisati kao:

$$-(1-P_b)\log_2[0.5(1-P_b)]=-(1-P_b-P_g)\log_2[0.5(1-P_b)]-P_g\log_2[0.5(1-P_b)].$$

Sada možemo grupisati po dva člana izraza za kapacitet, tako da dobijamo:

$$C=(1-P_b-P_g)\log_2\frac{2(1-P_g-P_b)}{1-P_b}+P_g\log_2\frac{2P_g}{1-P_b},$$

čime je dokaz okončan.

4.5. Data je tranziciona matrica binarnog kanala:

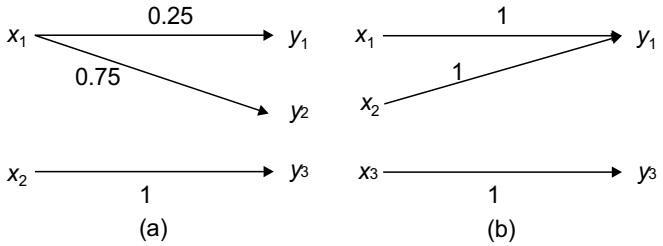
$$\mathbf{P}=\begin{bmatrix}2/3 & 1/3\\ 1/10 & 9/10\end{bmatrix}$$

Odrediti kapacitet ovakvog kanala.

Rješenje: U pitanju je nesimetrični binarni kanal kod kojeg je vjerovatnoća greške za jedan simbol 1/3, dok je za drugi simbol 1/10. Na osnovu formule koja je izvedena za slučaj binarnog nesimetričnog kanala, slijedi da je kapacitet jednak:

$$C=\log_2\left[1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}\right]-\frac{H(p_0)(1-p_1)-H(p_1)p_0}{1-p_0-p_1}.$$

Uvrštavajući $H(0.1)=0.469$, dok je $H(2/3)=0.9183$, dobijamo $C=0.2676$ bita/simbolu.



Slika IV.13. Modeli dva kanala

4.6. Za kanale koji su dati na Slici IV.13 odrediti tranzicionu matricu i odrediti kapacitet.

Rješenje: Tranzicione matrice ova dva kanala su:

$$\mathbf{P}_1 = \begin{bmatrix} 0.25 & 0.75 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{P}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Za prvi kanal:

Pretpostavimo da je vjerovatnoća pojavljivanja prvog simbola p, dok je vjerovatnoća pojavljivanja drugog simbola $1-p$. Vjerovatnoće pojavljivanja simbola na strani prijema su: $[0.25p, 0.75p, 1-p]$. Združene vjerovatnoće simbola na ulazu i izlazu su $[0.25p, 0.75p, 1-p, 0, 0, 0]$. Međusobna informacija je jednaka:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = H(X) = H(p).$$

Dakle, zaključujemo da je kapacitet ovog kanala $C = 1$ bit/simbolu.

Za drugi kanal:

Pod pretpostavkom da su vjerovatnoća prvog i drugog simbola p, q, slijedi da je vjerovatnoća trećeg $1-p-q$. Simboli na izlazu pojavljuju se sa vjerovatnoćama $p+q$ i $1-p-q$. Združena entropija ovih simbola su $[p, q, 1-p-q]$. Međusobna informacija u ovom slučaju je:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = H(Y) = H(p+q).$$

Ponovo je i u ovom slučaju kapacitet maksimalan za $p+q = 0.5$ i iznosi $C = 1$ bit/simbolu.

4.7. Na ulaz kanala moguće su poruke –1, 0 i 1. Kanal vrši kvadriranje ulazne poruke i nema brisanja, ni grešaka. Čemu je jednak kapacitet kanala?

Rješenje: Pretpostavimo da su vjerovatnoće simbola na ulazu, redom: $p, q, 1-p-q$. Mogući ishodi na izlazu su, pod datim uslovima, 0 (sa vjerovatnoćom q) i 1 (sa vjerovatnoćom $p+1-p-q=1-q$). Entropija na izlazu je, dakle, $H(Y)=H(q)$. Združena entropija je jednaka:

$$H(X,Y)=H(X)=-p\log_2 p-q\log_2 q-(1-p-q)\log_2(1-p-q).$$

Dakle, međusobna informacija je jednaka:

$$I(X;Y)=H(Y)=H(q),$$

pa je kapacitet kanala jednak $C=$ 1 bit/simbolu.

4.8. Može li binarni kanal sa brisanjem biti slabosimetričan? Obrazložiti odgovor. Ako može, pokazati da se kapacitet kanala za ovaj slabosimetrični kanal može dobiti putem formule za slabosimetrične kanale.

Rješenje: Binarni kanal sa brisanjem ima sljedeću matricu tranzicije:

$$\mathbf{P}=\begin{bmatrix}1-\alpha & \alpha & 0\\ 0 & \alpha & 1-\alpha\end{bmatrix}.$$

Druga vrsta matrice je permutacija prve vrste, ali da bi kanal bio slabosimetričan potrebno je da suma vjerovatnoća u kolonama bude konstantna, a to je moguće samo za $\alpha=1/3$. Kapacitet kanala u tom slučaju iznosi:

$$C=1-\alpha=2/3.$$

Sada primijenimo, na ovaj kanal, formulu za kapacitet slabosimetričnog kanala (u specijalnom slučaju za $\alpha=1/3$):

$$\begin{aligned}C&=\log_2 3-H(\mathbf{r})=\log_2 3+\alpha\log_2\alpha+(1-\alpha)\log_2(1-\alpha)=\\&=\log_2 3+\frac{1}{3}\log_2\frac{1}{3}+\frac{2}{3}\log_2\frac{2}{3}=\log_2 3-\frac{1}{3}\log_2 3+\frac{2}{3}\left[1-\log_2 3\right]=\\&=\frac{2}{3}+\log_2 3-\frac{1}{3}\log_2 3-\frac{2}{3}\log_2 3=\frac{2}{3}.\end{aligned}$$

Dakle, u ovom specijalnom slučaju može se primijeniti formula za kapacitet slabosimetričnog kanala. Slično važi i u slučaju kanala sa greškom i brisanjem.

4.9. Pretpostavimo da imamo kod sa kodnom riječju dužine 3, gdje je jedan bit informacioni. Kod je u stanju da ispravi jednu pogrešku. Razmotrite vjerovatnoću greške u kodu, u zavisnosti od vjerovatnoće greške u kanalu i uporedite sa rezultatima koje daje kodna teorema. Ponoviti postupak za slučaj kada imate kod sa 7 bita, od kojih je 4 informacionih, sa mogućnošću ispravke jedne pogreške. Zatim ponovite postupak ako imate kod sa 15 bita, od kojih

je 11 informacionih, sa mogućnošću ispravke jedne pogreške. Nakon toga obaviti proceduru ako imate pet bita, a samo jedan informacioni, ali kod može da ispravi do dvije pogreške. Ponoviti proceduru nad kodom koji ima 15 bita, od kojih je 7 informacionih i koji može da ispravi do 2 pogreške. Konačno, uzeti slučaj 7 bita, od kojih je jedan informacioni i koji može da ispravi tri pogreške, i koda sa 15 bita, od kojih su samo pet informacionih, ali kod ima mogućnost da ispravi do tri pogreške.

Rješenje: Kodni odnosi u navedenim slučajevima su: $R_{1,3} = 1/3$, $R_{4,7} = 4/7$, $R_{11,15} = 11/15$, $R_{1,5} = 1/5$, $R_{7,15} = 7/15$, $R_{1,7} = 1/7$ i $R_{5,15} = 5/15 = 1/3$, gdje indeksi kodnog odnosa označavaju broj informacionih i ukupan broj bita u kodu. Kapacitet binarnog simetričnog kanala je $C = 1 - H(p)$ i da bismo po II Šenonovoj teoremi imali vjerovatnoću greške koja teži nuli, treba da važi $R \le C$. Granične verzije ove vjerovatnoće su $p_{1,3} = 0.1739$, $p_{4,7} = 0.0876$, $p_{11,15} = 0.0454$, $p_{1,5} = 0.2430$, $p_{7,15} = 0.1214$, $p_{1,7} = 0.2812$ i $p_{5,15} = 0.1739$. U slučaju kodova koji mogu da isprave jednu, dvije i tri pogreške imamo da su vjerovatnoće pogreške u tim situacijama:

$$P_1 = 1 - (1-p)^n - np(1-p)^{n-1}$$

$$P_2 = 1 - (1-p)^n - np(1-p)^{n-1} - n(n-1)p^2(1-p)^{n-2}/2$$

$$P_3 = 1 - (1-p)^n - np(1-p)^{n-1} - n(n-1)p^2(1-p)^{n-2}/2 - n(n-1)(n-2)p^3(1-p)^{n-3}/6.$$

Za prvi kod u kritičnoj (graničnoj) situaciji zadovoljenja kodne teoreme imamo $P_1(n = 3, p_{1,3}) \approx 0.0802$ (u zagradama smo naznačili dužinu kodne riječi i graničnu vjerovatnoću). Za ostale kodove slijedi:

$P_1(n = 7, p_{4,7}) \approx 0.1196$ (veća vjerovatnoća greške kojom plaćamo bolji kodni odnos)

$$P_1(n = 15, p_{11,15}) \approx 0.1466$$

$P_2(n = 5, p_{1,5}) \approx 0.0963$ (veća vjerovatnoća greške u odnosu na prvi slučaj, ali kod ispravlja dvije pogreške)

$$P_2(n = 15, p_{7,15}) \approx 0.2714$$

$P_3(n = 7, p_{1,7}) \approx 0.1030$ (ispravlja tri pogreške, ali uz mali kodni odnos)

$$P_3(n = 15, p_{5,15}) \approx 0.2565.$$

Ove greške imaju dva uzroka: bitniji je onaj koji se odnosi na ograničenu dužinu kodne riječi (II Šenonova teorema je izvedena u asimptotskim uslovima), kojem podršku daje drugi problem, koji se ogleda u neoptimalnosti kodnog postupka.

Da bismo ovo još malo osvijetlili, pogledajmo šta se dešava kada se malo udaljimo od granice kodne teoreme, odnosno kada imamo da je vjerovatnoća greške po bitu jednaka polovini granične:

$$P_1(n=3,p_{1,3}/2)\approx 0.0214 \quad P_1(n=7,p_{4,7}/2)\approx 0.0348 \quad P_1(n=15,p_{11,15}/2)\approx 0.0445$$
$$P_2(n=5,p_{1,5}/2)\approx 0.0148 \quad P_2(n=15,p_{7,15})\approx 0.0588 \quad P_3(n=7,p_{1,7}/2)\approx 0.0096$$
$$P_3(n=15,p_{5,15}/2)\approx 0.0358.$$

Vidimo da dolazi do drastičnog smanjivanja vjerovatnoće greške u procesu dekodiranja u odnosu na situaciju kada radimo sa vjerovatnoćama greške po bitu koje su dostižne onima po II Šenonovoj teoremi.

4.10. Odrediti vjerovatnoće u stacionarnom stanju za Smit–Boven–Džojsov model kanala.

Rješenje: Matrica tranzicije ovog kanala (Slika IV.9) je:

$$\mathbf{P}=\begin{bmatrix} P_{00} & 0 & P_{0L} \\ 0 & P_{11} & P_{1L} \\ P_{L0} & P_{L1} & P_{LL} \end{bmatrix}.$$

Uslovne vjerovatnoće zadovoljavaju sljedeće relacije:

$$P_{00}+P_{\mathrm{L0}}=1 \qquad P_{11}+P_{L1}=1$$
$$P_{L0}+P_{L1}+P_{LL}=1,$$

pa se matrica tranzicije može svesti na:

$$\mathbf{P}=\begin{bmatrix} 1-P_{0L} & 0 & P_{0L} \\ 0 & 1-P_{1L} & P_{1L} \\ P_{L0} & P_{L1} & 1-P_{L0}-P_{L1} \end{bmatrix}.$$

Vjerovatnoće u stacionarnom stanju slijede iz jednačine:

$$\mathbf{(P\text{-}I)}^T\mathbf{p}=0,$$

uz zamjenu jedne od jednačina trivijalnom jednačinom da suma vjerovatnoća stanja mora biti jednaka 1 (pogledati Poglavlje II.2.5):

$$\begin{bmatrix} -P_{0L} & 0 & P_{L0} \\ 0 & -P_{1L} & P_{L1} \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} p(0) \\ p(1) \\ p(L) \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Konačno dobijamo:

$$p(0)P_{0L}=p(L)P_{L0} \qquad p(1)P_{1L}=p(L)P_{L1}$$
$$p(0)+p(1)+p(L)=p(L)\left[\frac{P_{L0}}{P_{0L}}+\frac{P_{L1}}{P_{1L}}+1\right]=1$$

$$p(L)=\frac{P_{0L}P_{1L}}{P_{L0}P_{1L}+P_{0L}P_{L1}+P_{0L}P_{1L}}$$

$$p(0)=\frac{P_{L0}p(L)}{P_{0L}}=\frac{P_{L0}P_{1L}}{P_{L0}P_{1L}+P_{0L}P_{L1}+P_{0L}P_{1L}}$$

$$p(1)=\frac{P_{L1}p(L)}{P_{1L}}=\frac{P_{0L}P_{L1}}{P_{L0}P_{1L}+P_{0L}P_{L1}+P_{0L}P_{1L}}.$$

IV.6.2 Softverska realizacija

A. Simulirati prenos sekvence od 10000 simbola ternarnog koda preko simetričnog kanala kod kojega je vjerovatnoća greške (uzeti u primjeru $P=0.05$) uniformno raspoređena na dva pogrešna simbola.

```
>> N=10000;
>> P=0.05;
>> Poruka=randi(3,1,N)-1;
>> R=rand(1,N);
>> Greska=(R<P/2)-(R>(1-P/2));
>> Primljeno=rem(Poruka+Greska+3,3);
```

Postupak je dosta jednostavan: kreiramo putem randi ternarnu poruku datih dimenzija. R je slučajni uniformni šum na intervalu [0,1]. Kada je šum ispod P/2 (to je vjerovatnoća 0.025), pravimo grešku koja je jednaka $+1$, a kada je R>0.975 (ponovo vjerovatnoća 0.025), pravimo grešku koja je jednaka -1. Sabiranjem Poruka i Greska po modulu 3 (sabiramo 3 da bismo izbjegli negativne vrijednosti), dobijamo primljenu poruku.

B. Kreirati binarni kanal sa greškom i brisanjem. Neka je vjerovatnoća greške $P_g=0.05$ i vjerovatnoća brisanja $P_b=0.01$. Stanje brisanja modelovati kao broj -1. Neka poruka ima dužinu $N=10000$ bitova.

```
>> N=10000;
>> Pg=0.05;
>> Pb=0.01;
>> Poruka=rand(1,N)>0.05;
>> R=rand(1,N);
>> Greska=(R<Pg);
>> Brisanje=(R>(1-Pb));
>> Primljeno=rem(Poruka+Greska,2).*(1-Brisanje)-Brisanje;
>> length(find(Primljeno==-1))
108
>> length(find(Primljeno~=Poruka))
607
```

Postupak je sličan prethodnom. Poruka je jednaka jedan kada je uniformni šum preko 1/2 (vjerovatnoća 0.5) i nula, u suprotnom. Greška se dobija na osnovu R kada je slučajna promjenljiva R ispod Pg, dok stanje brisanja dobijamo kada je R veće iznad (1-Pb) (vjerovatnoća P_b). Primljenu poruku formiramo tako što saberemo (po modulu 2) poruku i grešku za one pozicije gdje nema brisanja, dok je -1 na pozicijama gdje ima brisanja. Posljednje dvije naredbe daju broj gdje smo detektovali brisanje (u našoj realizaciji bilo je 108, što je približno 1%, odnosno P_b), odnosno broj pozicija na kojima se primljena poruka razlikuje od poslate (približno 6%, što je aproksimativno $P_b + P_g$).

C. Odrediti kapacitet kanala sa tranzicionom matricom:

$$\begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.05 & 0.15 & 0.8 \end{bmatrix}.$$

U pitanju je očigledno nesimetrični ternarni kanal, a vidjeli smo kod nesimetričnog binarnog kanala da je određivanje kapaciteta netrivijalno. Stoga smo odlučili da to uradimo numerički. Pretpostavićemo da se simboli na ulazu pojavljuju sa vjerovatnoćama p_1, p_2, $1-p_1-p_2$. Cilj nam je odrediti p_1 i p_2 tako da je kapacitet maksimalan. MATLAB kod koji ovo određuje (direktnom pretragom) je:

```
clear
K=[0.7 0.2 0.1; 0.3 0.6 0.1; 0.05 0.15 0.8];
Cmax=0;p1m=0;p2m=0;
for p1=0:0.0001:1
   for p2=0:0.0001:1
      [p1,p2]
      if(p1+p2<=1)
         p3=1-p1-p2;
         P1=p1*K(1,1)+p2*K(2,1)+p3*K(3,1);
         P2=p1*K(1,2)+p2*K(2,2)+p3*K(3,2);
         P3=p1*K(1,3)+p2*K(2,3)+p3*K(3,3);
         %%%Ovo su vjerovatnoce za Y pa je entropija HY
         HY=-P1*log2(P1)-P2*log2(P2)-P3*log2(P3);
         Hxy=-p1*sum(K(1,:).*log2(K(1,:)))-...
  p2*sum(K(2,:).*log2(K(2,:)))-p3*sum(K(3,:).*log2(K(3,:)));
         C=HY-Hxy;
         if(C>Cmax)
            Cmax=C;
            p1m=p1;
            p2m=p2;
         end
      end
   end
end
```

Programska realizacija je jednostavna, a daleko od optimalne. Usvajamo vjerovatnoće prva dva simbola, a zatim, na osnovu njih, računamo vjerovatnoću trećeg simbola kao $1 - p_1 - p_2$. Zatim računamo kolika je vjerovatnoća simbola na izlazu iz kanala (P1, P2 i P3), a to koristimo da bismo sračunali $H(Y)$, dok nam je za računanje $H(Y|X)$ dovoljna matrica tranzicije i vjerovatnoće p_1, p_2 i p_3. Kada sračunamo međusobnu informaciju, u slučaju da je veća od dotadašnjeg maksimuma, proglašavamo je tekućim maksimumom i pamtimo. Konačni rezultati su:

$$C = 0.4962,$$

koje se dobija za $p_1 = 0.348$ i $p_2 = 0.196$.

POGLAVLJE V

UVOD U KODIRANJE KANALA

V. UVOD U KODIRANJE KANALA

Već iz činjenice da je materija vezana za kodiranje kanala podijeljena u tri poglavlja, jasno je da je u pitanju složenija problematika od one koju smo izučavali kod kodiranja izvora. U ovom poglavlju, u Sekciji V.1, dajemo pregled osnovnih pojmova, pa prelazimo na kodove koji su u stanju samo da detektuju pogreške koje se dešavaju u kanalu. Obradili smo i binarne i nebinarne **kodove za detekciju pogreške**. Nakon toga, intuitivnim putem uvodimo osnovne **kodove za korekciju pogreške**: **repetitivni kod** (ponavljajući ili kod sa većinskom − majoritetnom logikom), **pravougaoni** i **trougaoni** kod. Nakon toga, u Sekciji V.3, uvodimo **Hemingov kod**. Zasigurno se može reći da su Šenonovi teorijski rezultati doživjeli poseban uspjeh na osnovu činjenice da su gotovo paralelno sa njihovim objavljivanjem nastali veoma kvalitetni kodovi za kodiranje izvora (Hafmenov kod) i kanala (Hemingov kod). U istoj sekciji opisali smo i neke varijante Hemingovog koda. Poglavlje zaključujemo sa pojmovima koji su bitni sa stanovišta performansi kodova: **Hemingovom distancom**, **Hemingovom težinom** i **minimalnim Hemingovim rastojanjem** (distancom) koda. Na ovaj način ćemo zaokružiti intuitivno uvođenje kodova za kodiranje kanala, a znatno bolje praktično i matematičko fundiranje biće dato u narednom poglavlju.

V.1 Tipovi kodova i detekcija pogreški

Kodove za kodiranje kanala možemo podijeliti u dvije klase: kodove za detekciju pogreške i kodove za korekciju pogreške. Kodovi za detekciju (engl. *error detecting codes*) pogreške u stanju su samo da detektuju da se pogreška dogodila, te da eventualno u komunikacijama zatraže ponovno slanje poruke ili barem obavijeste korisnika o problemu. Kodovi za korekciju pogreške (engl. *error correcting codes*) su znatno složeniji i važniji, a mogu da isprave pogrešku, automatski, bez intervencije korisnika. Postoje kodovi koji omogućavaju „miješane“ sposobnosti,

odnosno mogu da se koriste za obje funkcionalnosti. Na primjer, postoje popularni kodovi koji su u stanju da isprave jednu i detektuju dvije pogreške u kodnoj riječi. U ovoj sekciji biće opisani neki od najpopularnijih postupaka za detekciju pogreške.

Binarni zapis (kod) je definisan na alfabetu {0,1}, oktalni kod je definisan na alfabetu {0, 1, ..., 7}, heksadekadni kod je definisan na alfabetu {0, 1, 2, ..., 9, a, b, c, d, e, f}. Ovi kodovi imaju jedinstvenu jednostavnu međusobnu vezu i vezu sa dekadnim zapisom broja. Pored toga, dekadni brojevi se mogu predstavljati u BCD kodu. Zapis karaktera u računarskim sistemima obezbijeđen je preko ASCII tabele ili *unicode* sistema. Bez redundantnih elemenata (simbola), ovi kodovi nisu u stanju da omoguće detekciju i korekciju pogreški, već samo mogu biti jednoznačno dekodabilni jer se svi simboli međusobno razlikuju.

ASCII tabela predstavlja kôd kod kojeg svakom karakteru odgovara kodna riječ jednake dužine. Postoje kodovi kao što je Morzeov, kod kojeg su slova engleske abecede predstavljena crtama i tačkama, kodom promjenljive dužine (sa manje simbola se kodiraju oni karakteri koji se češće pojavljuju u jeziku). Morzeov kod je korišćen u telegrafskom prenosu poruka, a i danas se neke njegove varijante koriste za komunikaciju radio-putem. Poruke poslate bilo kojim od ovih kodova, putem komunikacionog kanala, podložne su greškama zbog uticaja smetnji, pa je neophodno dodati određenu redundanciju u kodnoj riječi kako bi se omogućio pravilan prijem (dekodiranje) poruka.

V.1.1 Detekcija pogreški kod binarnih kodova

ASCII kod se pojavljuje u dvije varijante. Obje varijante vrše zapis simbola sa 8 bita. U prvoj varijanti svih 8 bita nose poruku i na taj način postoji mogućnost prenosa $2^8 = 256$ različitih karaktera. Kako bi se omogućila detekcija pogreški, uvedena je druga varijanta. Kod ove varijante informaciju nosi 7 bita (nazivaju se **informacioni biti**), tako da se može kodirati $2^7 = 128$ različitih karaktera. Na kraju ove poruke 8. bit je tzv. **bit parnosti**, koji se bira tako da je ukupan broj jedinica u ovoj kodnoj poruci paran. Dakle, posljednji bit nije nezavisan, već je redundantan. Ako se u poruci primi paran broj jedinica, zaključujemo da je poruka uspješno prenesena, dok ako to nije slučaj, zaključujemo da je u poruci došlo do greške, te da njen sadržaj nije pouzdan. Stoga je ovakav **kod sa provjerom parnosti** najjednostavniji, a opet i veoma koristan, kako teorijski tako i praktično, za detekciju pogreške. Na osnovu njega možemo uvesti dosta bitnih koncepata i pojmova za praćenje ostatka materije vezane za kodiranje kanala.

Pretpostavimo da u opštem slučaju imamo k informacionih bita. Kako dodajemo jedan redundantni bit, ukupna dužina poruke je:

$$n = k + 1.$$

Kodni odnos ovog koda je:

$$R = \frac{\log_2 M}{n} = \frac{\log_2 2^k}{n} = \frac{k}{n} = \frac{k}{k+1},$$

gdje je M broj simbola koji se datim kodom kodiraju, u datom slučaju $M = 2^k$.

Kodni odnos $R = k/n$ (odnos broja informacionih s ukupnim brojem bita u kodnoj riječi) karakteristika je svih **blok kodova**. Pretpostavimo da je vjerovatnoća greške po bitu p i da je vjerovatnoća greške nezavisna na pojedinim simbolima. Tada je vjerovatnoća uspješnog prenosa (bez grešaka):

$$(1-p)^n.$$

Vjerovatnoća jedne pogreške je jednaka:

$$\binom{n}{1} p(1-p)^{n-1} = np(1-p)^{n-1}.$$

Međutim, naš kod je u stanju da detektuje pojavu svakog neparnog broja grešaka u riječi:

$$\sum_{i=1}^{\lceil n/2 \rceil} \binom{n}{2i-1} p^{2i-1}(1-p)^{n-2i+1},$$

gdje oznaka $\lceil \ \rceil$ predstavlja zaokruživanje na veći cijeli broj. Često kada je vjerovatnoća greške jako mala ($p \ll 1$), vjerovatnoća da se detektuje pogreška može se aproksimirati prvim članom u predmetnoj sumi:

$$\sum_{i=1}^{\lceil n/2 \rceil} \binom{n}{2i-1} p^{2i-1}(1-p)^{n-2i+1} \approx np(1-p)^{n-1}.$$

Postoji i mogućnost da kod ne može da posluži svojoj svrsi, odnosno da se dogode situacije da u kodnoj riječi postoje greške koje se ne mogu detektovati. U našem slučaju to je situacija kada se dešava paran broj pogreški:

$$\sum_{i=1}^{\lceil n/2 \rceil} \binom{n}{2i} p^{2i}(1-p)^{n-2i} \approx \binom{n}{2} p(1-p)^{n-1}.$$

Ponovo se ovaj izraz za realne situacije kada je vjerovatnoća greške u kanalu relativno mala može aproksimirati prvim članom u predmetnoj sumi.

Postoji još jedan zgodan koncept koji možemo uvesti već kod ovog „primitivnog“ kodnog sistema. Uslov parnog broja jedinica u kodnoj riječi može se zapisati na sljedeći način:

$$i_1 \oplus i_2 \oplus \cdots \oplus i_k \oplus c_{k+1} = 0,$$

gdje su $i_j, j = 1,...,k$ informacioni biti, dok je c_{k+1} bit parnosti. Kada budemo koristili kodne riječi, radi jednostavnosti, oznaka za ekskluzivno ili $\oplus$ biće zamijenjena simbolom +. Ovo smo već opisali u prvom poglavlju kada smo razmatrali koncept grupe. U ovom slučaju grupa je definisana preko binarnog alfabeta, a operacija sabiranja je ekskluzivno ili, a množenja logička konjukcija:

$$i_1 + i_2 + \cdots + i_k + c_{k+1} = 0.$$

Ekskluzivno ili operacija se ponekad naziva i sabiranje po modulu 2, pa je gornji izraz tačan kada u poruci imamo paran broj jedinica. Već smo vidjeli kod Grejovog koda da se ispunjenje ovog uslova može zapisati i kao:

$$c_{k+1} = i_1 \oplus i_2 \oplus \cdots \oplus i_k$$
$$c_{k+1} = i_1 + i_2 + \cdots + i_k.$$

Dakle, bit parnosti se jednostavno hardverski realizuje primjenom ekskluzivno ili kola. Provjera ispravnosti primljene riječi obavlja se operacijom ekskluzivno ili nad primljenom kodnom riječju i poređenjem sa logičkom nulom.

Sljedeća dva binarna koda za detekciju grešaka nazivaju se „2 od 5“ i „3 od 7“. Kod njih su kodne riječi dužine 5, odnosno 7, dok se u prvom slučaju jedan od simbola (uzmimo da je to jedinica) pojavljuje 2 puta, a u drugom slučaju se taj simbol pojavljuje 3 puta. Broj dozvoljenih kombinacija (ispravnih kodnih riječi) u ovim kodovima je:

$$M = \binom{5}{2} = 10 \qquad M = \binom{7}{3} = 35.$$

Kodni odnosi za ove kodove su:

$$R = \frac{\log_2 10}{5} \approx 0.6644 \qquad R = \frac{\log_2 35}{7} \approx 0.7328.$$

Prvi kod je pogodan za prenos velike količine numeričkih podataka pošto se sa deset simbola efikasno kodira deset cifara. Drugi kod može da predstavi 35 simbola. Pogodan je za predstavljanje slova, pa je stoga korišćen, čak i danas, u sistemima za prenos tekstualnih poruka putem radio-kanala (engl. *TOR – Telex Over Radio*), što se često koristi u pomorskim komunikacijama. Međutim, problem je u činjenici da ovakve poruke ne mogu da predstave cifre, interpunkciju i druge simbole od interesa. Stoga se prave dvije tabele simbola, od kojih jedna predstavlja prenos slova, a druga cifara i specijalnih simbola. Nakon prijema specijalnog karaktera, koji se naziva „prelaskom na slike“ (engl. *figure shift*) − često se koristi kod 0100110, primaju se cifre i specijalni simboli, dok je povratak na prijem slova označen specijalnim karakterom, koji se naziva „prelazak na slova“ (engl. *letter shift*) − česta

kombinacija je 0001110. Sistem u pomorskim komunikacijama funkcioniše tako da prima blok od tri karaktera (21 bit), pa ako nije detektovana pogreška, nastavlja se sa prijemom narednog bloka od 21 bita, dok u slučaju da je greška detektovana, prijemna strana traži da se blok ponovo pošalje.

Razmotrimo sada vjerovatnoću detekcije pogreške u slučaju koda „2 od 5“ (u dijelu urađenih zadataka razrađen je slučaj koda „3 od 7“). Vjerovatnoća da nema pogreške za slučaj kada je vjerovatnoća greške po bitu jednaka p i kada su greške na pojedinim bitima međusobno nezavisne jednaka je:

$$(1-p)^5.$$

Detekcija greške se dešava u svakom slučaju kada se pojavi neparan broj pogreški (jedna, tri i pet):

$$\binom{5}{1}p(1-p)^4+\binom{5}{3}p^3(1-p)^2+\binom{5}{5}p^5=5p(1-p)^4+10p^3(1-p)^2+p^5.$$

Međutim, ovdje postoji mogućnost detekcije i parnog broja pogreški. Naime, u slučaju 2 pogreške, ako se jedna desi na bitu sa vrijednošću 1, a druga na bitu sa vrijednošću 0, ukupan broj jedinica neće biti promijenjen i ova pogreška neće biti detektovana. Ako su obje pogreške na istim bitovima (dvije nule postale jedinice ili dvije jedinice pretvorene u nule), ovakva promjena će biti detektovana. U slučaju sa pogreškama na četiri bita, neće biti detektovane onda kada se dvije dogode na nulama, a dvije na jedinicama, jer tada neće doći do promjene broja jedinica u riječi, dok će ostale greške biti detektovane (greške na tri nule i jednoj jedinici i na svim nulama). Broj slučajeva kada imao pogreške na dva bita je $\binom{5}{2}=10$, dok je broj slučajeva sa pogreškama na četiri bita $\binom{5}{4}=4$. Od 10 slučajeva sa dvije pogreške, u četiri prepoznajemo grešku $\binom{3}{2}+\binom{2}{2}=4$ (tri slučaja sa dvije promjene na nultim bitovima i jedan sa greškama na jediničnim bitovima), dok od pet slučajeva, kada se dese četiri pogreške, dva detektujemo, a to su slučajevi grešaka na sve tri nule i jednoj jedinici. Dakle, u slučaju koda „2 od 5“, vjerovatnoća detektovanih pogreški je:

$$5p(1-p)^4+4p^2(1-p)^3+10p^3(1-p)^2+2p^4(1-p)+p^5,$$

dok je vjerovatnoća nedetektovanih pogreški:

$$6p^2(1-p)^3+3p^4(1-p).$$

U praksi se i ovdje često ograničavamo na prve članove u predmetnim sumama pošto je najčešće vjerovatnoća greške relativno mala, pa su ostali članovi znatno manji u odnosu na prvi član. Na Slici V.1 prikazane su vjerovatnoće kada u kodu ne postoji greška (plave linije), kada se greška detektuje (zelene linije) i kada se greška ne detektuje (crvene linije) za slučajeve ASCII koda, sa provjerom parnosti za $n=8$ bita (pune linije) i za kod „2 od 5“ (isprekidane linije). Činjenica je da se ove vjerovarnoće jasno vide kada je p relativno veliko (recimo, $p>0.04$) (Slika V.1(a)), ali je obično interesantniji dio za malo p. Stoga se predmetne vjerovatnoće često prikazuju u tzv. log-log dijagramima, gdje se vjerovatnoće i po x i po y osi prikazuju logaritamski (obično radi jasnoće, sa dekadnim logaritmom). Takav dijagram je prikazan na Slici V.I(b), gdje se, na primjer, sada jasno vidi da je vjerovatnoća nedetektovane pogreške za $p=0.0001=10^{-4}$ reda veličine 10^{-7}, dok je vjerovatnoća detektovane pogreške za $p=0.001=10^{-3}$ reda veličine 10^{-2}. Očigledno su ova dva načina prikaza kompatibilni, odnosno prvi je pogodniji za veće vjerovatnoće p, dok je drugi znatno pogodniji kada je ova vjerovatnoća realističnija (manja).

V.1.2 Detekcija pogreški kod nebinarnih kodova

Postoji potreba da se greška detektuje i kod nebinarnih kodova. Ova potreba je od velike praktične važnosti. Ilustrujmo je na primjeru zapisa knjiga.

Za jedinstvenu identifikaciju knjiga koristi se međunarodni standardni broj knjige (engl. *International Standard Book Number* – ISBN). U osnovnoj varijanti, on se sastoji od 10 cifara, od kojih su devet informacione, a deseta je kontrolna. Prvih devet cifara je obično grupisano u tri grupe, odvojene tireima, koje redom označaju: državu izdavača, samog izdavača, redni broj knjige. Posljednja se formira tako što se određuje odgovarajuća provjera. Pretpostavimo da je ova deseta cifra formirana tako da se dobije djeljivost sa 11 (kod većine nebinarnih kodova za detekciju pogreške koristi se provjera vezana za djeljivost sa prostim brojevima). Uzmimo primjer:

$$82\text{-}7539\text{-}467\text{-}c,$$

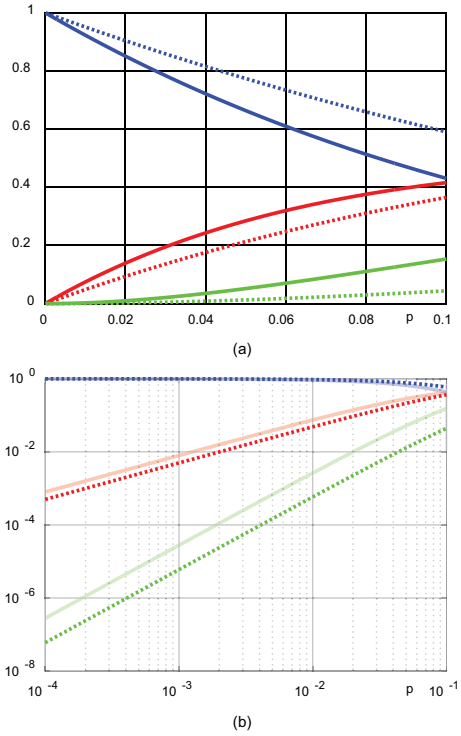
gdje je c kontrolna cifra. Suma ove riječi je jednaka

$$8+2+7+5+3+9+4+6+7+c=51+c.$$

Da bi se zadovoljila djeljivost sa 11, potrebno je da je $c=4$. Međutim, ovakav proces ne rješava sljedeći problem. Pretpostavimo da je permutovana treća i četvrta cifra:

$$82\text{-}\underline{\mathbf{57}}39\text{-}467\text{-}4.$$

Detekcija će reći da je ovakva riječ ispravna. Permutovanje cifara je čest problem u ručnom radu, a često i uređaji za automatsko čitanje vrše slične permutacije.



Slika V.1. Vjerovatnoća prenosa bez greške (plavo), detektovane pogreške (crveno), nedetektovane greške (zeleno) za kod sa provjerom parnosti (ASCII – puna linija) i kod „2 od 5“ (isprekidana): (a) linearna skala; (b) logaritamska skala po obje ose

Stoga je lako zaključiti da nebinarni kodovi za detekciju pogreške moraju biti u stanju da detektuju većinu permutacija koje se dogode (posebno one na susjednim simbolima). Ovaj problem se prevazilazi pridruživanjem različitih težina (multiplikativnih faktora) svakom kodnom simbolu. Označimo sada riječ koja predstavlja ISBN, kao:

$$i_1\, i_2\, i_3\, i_4\, i_5\, i_6\, i_7\, i_8\, i_9\, c.$$

Određivanje kontrolnog simbola (cifre) se obavlja tako da

$$\sum_{j=1}^{9} w_j i_j + c$$

bude djeljivo sa 11. Težinski koeficijenti $w_j, j \in [1,9]$ biraju se tako da budu različiti za susjedne simbole (radi otkrivanja permutacija simbola). U programerskoj notaciji, uslov djeljivosti sa prostim brojem 11 može se označiti kao:

$$\left(\sum_{j=1}^{9} w_j i_j + c\right)\%11 = 0,$$

gdje je % operator ostatka pri dijeljenju u više popularnih programskih jezika. Alternativno se za ovo mogu koristiti oznake rem(*a,p*) (rem od *reminder* – ostatak) ili mod(*a*,*p*). Kod ISBN broja težinski koeficijenti se biraju redom: 10, 9, 8,…, 2, dok je uz *c*, kao što vidimo, težinski koeficijent 1. Ovo se može zapisati i kao: $w_j = 11 - j,\ \ j \in [1,10]$. Kod posmatranog koda:

$$8\times 10 + 2\times 9 + 7\times 8 + 5\times 7 + 3\times 6 + 9\times 5 + 4\times 4 + 6\times 3 + 7\times 2 + c = 300 + c.$$

U ovom slučaju je $c = 8$ da bi obezbijedilo djeljivost sa 11 (308/11 = 28 bez ostatka). Postavlja se pitanje: šta ako se desi da je potreban ostatak $c = 10$, što je realna mogućnost kod djeljivosti sa 11? Po ISBN kodu, u tom slučaju se dodaje specijalni simbol $c = \mathrm{X}$. Ako se desi permutacija simbola, npr. na pozicijama l i $l+1$, onda, umjesto članova u sumi $w_l i_l + w_{l+1} i_{l+1}$, imamo $w_l i_{l+1} + w_{l+1} i_l$, to jeste kao da smo sumi dodali $w_l i_{l+1} + w_{l+1} i_l$ i oduzeli $w_l i_l + w_{l+1} i_{l+1}$:

$$\left(\sum_{j=1}^{9} w_j i_j + c\right) + w_l i_{l+1} + w_{l+1} i_l - w_l i_l - w_{l+1} i_{l+1}$$

$$= \left(\sum_{j=1}^{9} w_j i_j + c\right) + w_l (i_{l+1} - i_l) + w_{l+1} (i_l - i_{l+1}) =$$

$$= \left(\sum_{j=1}^{9} w_j i_j + c\right) + (w_l - w_{l+1})(i_{l+1} - i_l).$$

Prvi član u posljednjem izrazu je djeljiv sa 11, po načinu formiranja ISBN koda, pa je onda ostatak pri dijeljenju moguće dobiti samo na osnovu drugog člana $(w_l - w_{l+1})(i_{l+1} - i_l)$. Kako nas interesuju samo situacije kada su susjedni informacioni simboli različiti $i_{l+1} \neq i_l$, te kako su težine pridružene susjednim simbolima različite $w_l \neq w_{l+1}$, jasno je da se djeljivost sa prostim brojem 11 ovog proizvoda može dobiti samo ako je proizvod:

$$(w_l - w_{l+1})(i_{l+1} - i_l)\%11 = 0.$$

Pošto je 11 prost broj, to znači da nema proizvoda koji će dati djeljivost sa 11 osim ako je jedan od činilaca djeljiv sa 11. Međutim, ovo nije moguće jer razlika dvije različite cifre ne može, po apsolutnoj vrijednosti, biti veća od 9:

$$w_l - w_{l+1} = -1, \qquad 0 < |\, i_{l+1} - i_l \,| \leq 9.$$

Danas se umjesto ovog koda koristi njegova varijanta, koja se naziva ISBN-13. Kod ISBN-13 (ušao u upotrebu 2007. godine) podaci o izdanju su u prvih 12

cifara, dok je posljednja cifra kontrolna. Težine simbola na neparnim pozicijama su 1, a na parnim pozicijama su 3. Posljednja cifra se dodaje tako da suma bude djeljiva sa deset (dosta je rijedak slučaj da se provjerava djeljivost brojem koji nije prost). Ovo dovodi i do određenih problema, odnosno do nemogućnosti da se svaka greška na susjednim simbolima detektuje. Vidimo da je razlika težina na uzastopnim simbolima jednaka ± 2. Dakle, u slučaju da dođe do permutacije cifara na susjednim mjestima, imamo:

$$(w_l - w_{l+1})(i_{l+1} - i_l) = \pm 2(i_{l+1} - i_l),$$

a ovaj izraz može biti djeljiv sa 10 ako $(i_{l+1} - i_l) = \pm 5$. Konverzija iz ISBN u ISBN-13 obavlja se tako što se na početku postavi „978“, a kontrolna cifra se ponovo sračuna u skladu sa pravilama za ISBN-13.

Posljednji nebinarni kod za detekciju greške koji će ovdje biti uveden jeste jedinstveni matični broj građana (JMBG). Ovo je, mora se priznati, zastarjeli način jedinstvenog identifikovanja građana, koji je 1980-ih godina uveden u bivšoj Jugoslaviji. Do danas se zadržao u većini zemalja nastalih na ovom području, ali će, zbog problema privatnosti, vjerovatno biti postepeno napuštan.

Sastoji se od 13 cifara, grupisanih u četiri grupe:

ddmmggg-op-brj-c.

Ovdje smo dodali crtice kako bismo razdvojili pojedine grupe. Prvih 7 cifara su datum rođenja, dvije cifre za dan, dvije za mjesec i tri za godinu, jer se prva od godine izostavlja. Naredne dvije se odnose na šifru opštine u kojoj je dijete rođeno, npr. op = 21 je Podgorica. Generalno, sve opštine u Crnoj Gori su iz treće desetice (21−29). Naredne tri cifre su jedinstvene za datog čovjeka: 000−499 za muškarce i 500−999 za žene. Posljednja cifra je kontrolna. Kontrolna cifra se kreira tako što su prvih 6 težinskih koeficijenata u opadajućem redosljedu od 7 do 2, pa su zatim narednih šest cifara ponovo sa opadajućim težinama od 7 do 2, dok je kontrolna cifra bez težine. Kontrolna cifra se formira tako da težinska suma bude djeljiva sa 11. Ako je potrebno da se doda broj 10, kao kontrolna cifra u JMBG se postavlja cifra 0.

Kod JMBG potrebno je da je zadovoljeno:

$$\left[\sum_{l=1}^{6}(7-l)i_l + \sum_{l=1}^{6}(7-l)i_{6+l} + c\right]\%11 = \left[\sum_{l=1}^{6}(7-l)(i_l + i_{6+l}) + c\right]\%11 = 0.$$

Prethodni iskaz važi za $c \in [0,9]$, dok kada:

$$\left[\sum_{l=1}^{6}(7-l)(i_l+i_{6+l})\right]\%11=1,$$

gdje bi c trebalo biti 10, usvajamo da je $c=0$. Dakle, možemo zapisati kao:

$$c=\left[11-\left[\sum_{l=1}^{6}(7-l)(i_l+i_{6+l})\right]\%11\right]\%10.$$

Pitanje je da li u ovom slučaju postoji mogućnost pogreške u detekciji permutovanih cifara. Pošto je 11 prost broj, ovakva greška nije moguća za slučaj kada $\left[\sum_{l=1}^{6}(7-l)(i_l+i_{6+l})\right]\%11\neq 1$. Ostaje jedino mogućnost da se to dogodi kada je ostatak pri dijeljenju sume

$$\left[\sum_{l=1}^{6}(7-l)(i_l+i_{6+l})\right]\%11=1.$$

U ovom slučaju, permutovanje bilo koje od susjednih cifara koje se nalaze na pozicijama od prve do šeste, odnosno od sedme do dvanaeste, dovodi do promjene u sumi na sljedeći način:

$$\left[\sum_{l=1}^{6}(7-l)(i_l+i_{6+l})\right]+(w_k-w_{k+1})(i_{k+1}-i_k),$$

gdje su permutovane cifre na pozicijama k i $k+1$. U predmetnim slučajevima važi: $w_k-w_{k+1}=1$, pa nije moguće postići da $i_{k+1}-i_k$ bude jednako 10, čime se jedino može obezbijediti da cifra nula bude kontrolna. Dakle, jedina situacija kada kod JMBG može doći do nedetekcije pogreške kod permutovanja susjednih simbola je između šeste i sedme pozicije, sa težinama $w_6=2$ i $w_7=7$, gdje je $w_6-w_7=-5$, pa je za slučaj kada je $i_7-i_6=-2$ moguće da imamo problem sa nedetekcijom jedne permutacije na susjednim simbolima. Posmatrajmo jedan krajnje generički kod:

000004200006-0.

Kontrola kaže da je $4\cdot 2+2\cdot 7+2\cdot 6=34$, tako da je za potrebe održavanja djeljivosti sa 11 potrebno da dodamo kontrolnu cifru 10, odnosno u slučaju JMBG koda 0. Zamislimo da je došlo do permutacije šeste i sedme cifre:

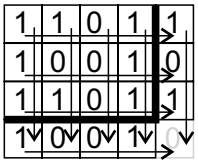
000002400006-0.

Provjera kaže da je $2\cdot 2+4\cdot 7+2\cdot 6=44$, odnosno da je kontrolna cifra jednaka 0, pa bi provjera sugerisala da je navedena kodna riječ ispravna. Naravno, ovim se ne iscrpljuju svi mogući tipovi grešaka koji se mogu desiti kod ovih kodova, ali smo se, kao što je već prikazano, ograničili samo na permutaciju susjednih

cifara, što je u osnovi bio i motiv za uvođenje gotovo svih nebinarnih kodova za detekciju pogreške.

V.2 Intuitivno uvođenje kodova za korekciju greške

Najjednostavniji kod za detekciju pogreške koji smo uveli bio je kod sa provjerom parnosti. Detekcija potvrđuje da u nekom nizu (vektoru) duž jedne dimenzije postoji greška u kodnoj riječi (preciznije, neparan broj grešaka; međutim, radi daljeg izvođenja, pretpostavljamo da postoji samo jedna greška). Ovaj kod može na zgodan način da posluži kao osnova za korekciju jedne pogreške. Naime, možemo formirati veći broj kodnih riječi, svaku sa bitom parnosti postavljenim na posljednjoj poziciji u kodnoj riječi. Poređajmo ove kodne riječi u tabelu tako što je svaka pojedinačna kodna riječ jedan red ove tabele. Dodajmo zatim još jednu riječ u posljednji red (vrstu) tabele, sa bitima koji predstavljaju provjeru parnosti u odgovarajućoj koloni tabele. Na Slici V.2 prikazana je ovakva šema koda koji se naziva **pravougaoni** iz jasnih razloga što su bitovi informacije i provjere parnosti postavljeni u oblik pravougaonika. Debelom linijom, na slici, ograđeni su informacioni biti, a posljednja kolona predstavlja provjere parnosti po vrstama, dok posljednja vrsta predstavlja provjere parnosti po kolonama. Posljedni bit se može izostaviti mada ćemo ga mi koristiti. Vidimo da imamo 12 informacionih bita, $k = 12$ (k je oznaka za broj informacionih bita), dok je ukupan broj bita u kodnoj tabeli (riječi) $n = 19$ (ako se uzme u obzir i posljednji, posivljeni bit, ima ih $n = 20$). Oznaka n se uobičajeno koristi u teoriji kodova za broj bita u kodnoj riječi. Ovakav kod pripada klasi blok kodova, koji se obično označavaju kao (n,k). Broj bita parnosti (redundantnih bita) obično se označava kao $m = n - k$. Što se tiče posljednjeg bita (u tabeli prikazano posivljeno), on se, kao što smo rekli, može izostaviti, mada, s druge strane, može da donese i neke dodatne elemente. Uglavnom će biti korišćen u knjizi. On se može tretirati kao bit provjere parnosti i po posljednjoj koloni i po posljednjoj vrsti. Kada ga postavimo, sve vrste i sve kolone matrice imaju paran broj jedinica, tako da se ovaj bit parnosti može tretirati i kao bit parnosti za čitavu tabelu. Može se pokazati da je ovo bit parnosti za sve informacione bite. Naime, ukupan broj jedinica u bitima parnosti u posljednjoj koloni (kada izuzmemo posljednji bit) je paran (neparan) ako je ukupan broj



Slika V.2. Ilustracija pravougaonog koda

jedinica među informacionim bitima paran (neparan). Isto važi i za posljednju kolonu. Stoga je ukupan broj bita parnosti u posljednjoj vrsti i koloni (isključujući posljednji bit) uvijek paran, tako da je provjera parnosti za čitavu tabelu jednaka provjeri parnosti samo informacionih bita!

Pretpostavimo da su dimenzije pravougaonika informacionih bita $k = k_1 \times k_2$; tada je ukupan broj bita u kodnoj riječi jednak $n = (k_1 + 1) \times (k_2 + 1)$. Kodni odnos ovakvog blok koda jednak je odnosu broja informacionih sa ukupnim brojem bita u kodnoj riječi:

$$R = \frac{k_1 k_2}{(k_1 + 1)(k_2 + 1)}.$$

Broj redundantnih bita potreban da bi se ispravila jedna pogreška je:

$$m = n - k = (k_1 + 1)(k_2 + 1) - k_1 k_2 = k_1 + k_2 + 1.$$

Dimenzije ovakog pravougaonog koda su $((k_1 + 1)(k_2 + 1), k_1 k_2)$.

Primjer V.1. Posmatrajmo 12 informacionih bita 101001111011. Kodirajmo ovu poruku pravougaonim kodom (9,4), a zatim u dobijeni tok podataka unesimo greške na treću, četrnaestu i dvadesetu poziciju i obavimo dekodiranje.

Rješenje: Tok podataka koji predstavlja poruku dijeli se u blokove (odatle i naziv blok kod) po onoliko bita koliko je informacionih u kodnoj poruci (po $k = 4$ u konkretnom primjeru):

1010 0111 1011.

Izvršimo sada kodiranje ove tri četvorke bita:

1	0	1
1	0	1
0	0	0

0	1	1
1	1	0
1	0	1

1	0	1
1	1	0
0	1	1

Sada je stvar neke usvojene konvencije kojim redom ćemo slati bitove kodnih riječi u kanal. Uzmimo da ih šaljemo red po red svake od tabela koje predstavljaju kodne riječi:

101101000 011110101 101110011.

Ponovo smo uveli male razmake na kraju svake kodne riječi isključivo radi vizuelnih efekata, a oni ne postoje u praksi. U uslovima zadatka dato je da su se greške dogodile na trećoj, četrnaestoj i dvadesetoj poziciji u kodu, tako da je riječ koja je primljena na strani dekodera (bite sa greškama smo vizuelno izdvojili od ostalih):

10**0**101000 0111**0**0101 1**0**1110011.

Ako dekodiranje vršimo putem tabela (što je trenutno jedino i moguće), primljene riječi transfomišemo u tri tabele:

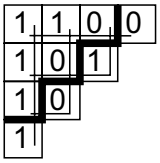
1	0	**0**
1	0	1
0	0	0

0	1	1
1	**0**	0
1	0	1

1	**1**	1
1	1	0
0	1	1

U drugoj tabeli prva vrsta i prva kolona ukazuju da u njima nije došlo do pogreške, dok u drugoj vrsti i drugoj koloni imamo pogrešku, pa zaključujemo da se greška dogodila na presjeku druge vrste i druge kolone. Kod binarnih kodova imamo olakšavajuću okolnost da kada znamo poziciju gdje se greška dogodila znamo i da je ispravimo, jer je u tom slučaju ispravni bit negacija primljenog bita. U trećoj tabeli situacija je slična, odnosno imamo indikaciju da se greška dogodila na presjeku prve vrste i druge kolone, pa na isti način ispravljamo primljenu poruku. Vraćamo se na prvu tabelu. U njoj je detektovana greška u prvoj vrsti, ali nije detektovana greška ni u jednoj od dviju kolona koje korespondiraju informacionim bitima. Međutim, na osnovu posljednje kolone, možemo uočiti da se greška dogodila na bitu parnosti u prvoj vrsti, ili na osnovu bita parnosti za čitavu tabelu zaključujemo da nema grešaka na informacionim bitima. Kako su ovaj i mnogi kodovi, koji će biti uvedeni, dizajnirani za slučaj jedne pogreške u kodnoj riječi, pretpostavićemo da nema više od jedne pogreške u riječi. Pokušajte sami da analizirate šta se dešava ako se pojavi više od jedne greške u predmetnom kodu.□

Ovdje se radi o očigledno jednostavnom, krajnje intuitivnom postupku, ali nas muči pitanje da li je ovaj postupak optimalan i kako bismo tu optimalnost postupka mogli da izmjerimo. U slučaju koda (9,4) imamo četiri informaciona bita i pet redundantnih sa kodnim odnosom $R = 4/9$, odnosno pet devetina bita gubimo u poruci. Naš cilj je da smanjimo redundanciju. Smanjivanje redundancije daje dvije koristi: smanjuje nepotrebnu upotrebu memorijskih resursa ili komunikacionog kanala (ekonomski gubici) i u dužim porukama (sa većim brojem redundantnih bita) veća je vjerovatnoća pojave više pogreški. Posmatrajmo sada situaciju kada imamo šest informacionih bita u bloku. Kod pravougaonog koda možemo ih rasporediti u pravougaonik dimenzija 2×3, a to dalje znači da kada dodamo kontrolne bite provjere parnosti, imamo pravougaonik dimenzija $3 \times 4 = 12$ bita, tako da je ovaj kod (12,6) i ima redundanciju od šest bita i kodni odnos $R = 1/2$.



Slika V.3. Ilustracija trougaonog koda

Isti efekat u odnosu na mogućnost ispravljanja jedne pogreške za šest informacionih bita možemo postići formirajući od ovih bita trougao sa redovima od po tri, dva i jednog informacionog bita, pa dodajući bite parnosti. Kod predmetnog **trougaonog koda** biti parnosti se nalaze na hipotenuzi pravouglog trougla. Predmetna struktura je grafički prikazana na Slici V.3. Bitovi parnosti, odvojeni od informacionih bita debelom linijom, provjeravaju svoju vrstu i kolonu. Kao što vidimo, prvi informacioni bit provjerava prvi red, drugi informacioni bit provjerava drugi red i treću kolonu, treći informacioni bit treći red i drugu kolonu, dok posljednji provjerava prvu kolonu.

Dakle, na predmetni način, svaki informacioni bit provjeravan je sa dva bita parnosti, tako da se nedvosmisleno može utvrditi njegova koordinata. Bitovi parnosti su provjeravani samo jednom. Kada jedna provjera parnosti nije zadovoljena, došlo je do greške na bitu parnosti. Uočimo da smo u predmetnom slučaju ostvarili mogućnost ispravke jedne pogreške na osnovu samo četiri redundantna bita (bita parnosti), dok smo u slučaju prethodnog pravougaonog koda to ostvarili sa šest redundantnih bita. Čak kada i isključimo posljednji bit parnosti kod pravougaonog koda i dalje imamo uštedu od jednog bita kod trougaonog. Predmetni trougaoni kod je blok kod (10,6), sa kodnim odnosom $R = 6/10 = 0.6$.

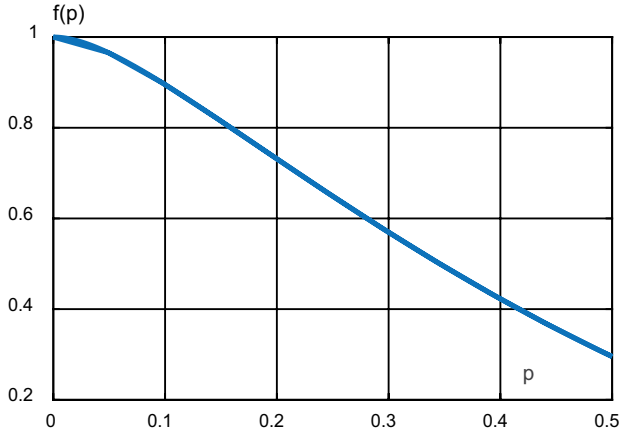
Primjer V.2. Uporediti vjerovatnoće ispravnog funkcionisanja trougaonog koda (10,6) i pravougaonog (12,6) pod pretpostavkom da je vjerovatnoća greške po bitu p, te da su greške i.i.d. procesa (nezavisne i na isti način distribuirane).

Rješenje: Vjerovatnoća da nema greške u kodnoj riječi kod ova dva koda je $(1-p)^n$, s tim da je kod prvog (trougaonog) koda $n = n_t = 10$, dok je kod drugog (pravougaonog) $n = n_p = 12$. Pošto oba koda imaju mogućnost da isprave jednu pogrešku, i ta situacija se može smatrati vjerovatnoćom uspješnog funkcionisanja koda. Vjerovatnoća da se to dogodi je $np(1-p)^{n-1}$. Dakle, ukupna vjerovatnoća da kodovi rade ispravno je:

$$(1-p)^n + np(1-p)^{n-1} = (1-p)^{n-1}[1-p+np] = (1-p)^{n-1}[1+(n-1)p].$$

Formirajmo odnos vjerovatnoća ispravnog funkcionisanja za predmetna dva koda:

$$f(p) = \frac{(1-p)^{11}[1+11p]}{(1-p)^9[1+9p]} = (1-p)^2\,\frac{1+11p}{1+9p}\,.$$



Slika V.4. Funkcija $f(p)$ odnosa pravilnog funkcionisanja pravougaonog i trougaonog koda u funkciji vjerovatnoće greške po bitu p

Slika V.4 demonstrira funkciju $f(p)$ u zavisnosti od vjerovatnoće greške i vidimo da je ona, kako raste p, znatno niža od 1, odnosno da je kod pravougaonog koda znatno manja vjerovatnoća pravilnog funkcionisanja nego kod trougaonog koda. Tako je za $p = 0.1$ vjerovatnoća pravilnog funkcionisanja pravougaonog koda približno 0.66, dok je kod trougaonog koda približno 0.736, te je funkcija $f(p) \approx 0.8967$. Situacija u korist trougaonog koda znatno se popravlja kako p raste.□

Primjer V.3. Povorka informacionih bita 1100101001001101111 kodirana je trougaonim kodom (10,6). Kodne riječi su formirane red po red odgovarajućeg trougla. Greške su se dogodile na pozicijama 6, 17, 21 i 23 u rezultujućem toku koji je poslat u kanal. Dekodirati poruke i tumačiti šta se dogodilo u procesu dekodiranja.

Rješenje: Prikažimo informacione bite i odgovarajuće bite parnosti:

1	1	0	0
0	1	1	
0	0		
1			

1	0	0	1
1	0	1	
0	0		
0			

1	1	0	0
1	1	0	
1	1		
1			

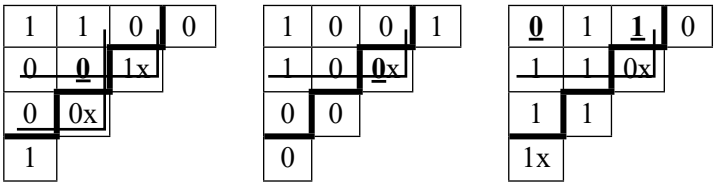
Dobijena poruka je (vizuelno razdvajamo riječi bjelinama radi bolje preglednosti):

1100011001 1001101000 1100110111.

U kanalu su generisane greške, pa je primljena poruka (greške su naglašene boldirano i podvučeno):

11000**0**1001 10011**0**0000 **0**1**1**0110111.

Prikažimo primljene poruke u obliku (trougaonih) tabela:



U prvoj tabeli imamo dva bita parnosti koja ukazuju na pogrešku i u presjeku linija koje kontrolišu ove provjere parnosti nalazi se bit na kojem se greška dogodila, pa je moguće izvršiti odgovarajuću ispravku. U drugoj tabeli samo jedan bit parnosti ukazuje da se greška u prenosu dogodila na bitu parnosti, tako da su svi informacioni biti ispravni. U trećem slučaju (kada u kodnoj riječi postoje dvije pogreške) ponovo dva bita parnosti ukazuju na pogreške, ali se u ovom slučaju u presjeku linija, na kojima se vrše provjere parnosti, nalazi pogrešan bit, pa trougaoni kod ne može da ispravi dvije greške (a tome nije ni namijenjen), već prilikom dekodiranja pravi dodatnu grešku. Dekodirana poruka je u ovom slučaju (vizuelno smo naznačili pozicije na kojima postoje greške):

$$110010 \quad 100100 \quad \mathbf{0}11\mathbf{1}011.$$

Postavlja se pitanje: šta bi se desilo s trougaonim kodom da se dvije greške nisu dogodile u istoj vrsti ili koloni, odnosno na istoj liniji koja predstavlja provjeru parnosti?□

Bitovi parnosti oba posmatrana blok koda, odnosno i kod pravougaonog i kod trougaonog koda, mogu se povezati s informacionim bitima jednostavnim algebarskim relacijama. Na primjer, posmatrajmo pravougaoni kod (9,4):

$$\begin{array}{lll} i_1 & i_2 & c_1 \\ i_3 & i_4 & c_2 \\ c_3 & c_4 & c_5. \end{array}$$

Prvi bit parnosti provjerava informacione bite i_1 i i_2, tako da, ako nema pogreški u kodnoj riječi, važi relacija:

$$i_1 \oplus i_2 \oplus c_1 = 0.$$

Sada kada saberemo sa c_1 i lijevu i desnu stranu jednakosti, dobijamo:

$$c_1 = i_1 \oplus i_2.$$

Podrazumijevaći da je u pitanju odgovarajuća algebra, u konačnom polju sa $P = 2$ elemenata, bez gubitka matematičke preciznosti, možemo pisati:

$$c_1 = i_1 + i_2.$$

Slične relacije se mogu uspostaviti i za ostale bitove parnosti (kontrolne bite):

$$c_2 = i_3 + i_4$$
$$c_3 = i_1 + i_3$$
$$c_4 = i_2 + i_4$$
$$c_5 = i_1 + i_2 + i_3 + i_4.$$

Slično možemo zapisati za bilo koji drugi blok kod. Na primjer, za već korišćeni trougaoni kod (10,6), s informacionim i kontrolnim bitima raspoređenim na sljedeći način:

$$\begin{matrix} i_1 & i_2 & i_3 & c_1 \\ i_4 & i_5 & c_2 & \\ i_6 & c_3 & & \\ c_4, & & & \end{matrix}$$

važi:

$$c_1 = i_1 + i_2 + i_3$$
$$c_2 = i_3 + i_4 + i_5$$
$$c_3 = i_2 + i_5 + i_6$$
$$c_4 = i_1 + i_4 + i_6.$$

V.3 Hemingovi kodovi – uvod

Postavlja se pitanje: da li je moguće konstruisati sistematski postupak za kodiranje informacione riječi date dužine tako da je broj kontrolnih bita koji omogućavaju ispravljanje jedne pogreške u kodnoj riječi minimalan? Minimalnost broja kontrolnih bita (parnosti) znači minimalnu redundanciju, odnosno minimalne dodatne memorijske zahtjeve (što utiče kako na smještaj podataka tako i na prenos podataka). Ujedno, kraće kodne riječi nam daju i minimalnu vjerovatnoću da se u nekoj kodnoj riječi generiše više od jedne pogreške, odnosno da se pogreška ne može ispraviti. Traženi postupak postoji i uveo ga je Ričard Heming (engl. *Richard Hamming*) u godinama koje su slijedile nakon završetka Drugog svjetskog rata, a paralelno s teorijskim doprinosima Kloda Šenona.

Posmatrajmo poruku od k informacionih bita. Na njih je potrebno dodati m redundantnih bita za provjeru parnosti tako da, na osnovu nekog postupka, uvijek možemo odrediti na kojoj od $n = m + k$ pozicija se dogodila pogreška. Dakle, m bita parnosti treba da identifikuje jednu pogrešku koja se može dogoditi na bilo kojem od n bitova (bilo informacionih ili bitova parnosti). Sa m bita se može identifikovati 2^m različitih obrazaca, ali jedan mora da bude sačuvan da bi predstavljao situaciju kada nema pogreški. Dakle, veza između broja redundantnih bita i dužine poruke trebala bi da bude:

$$2^m - 1 = n.$$

Odnosno, izraženo preko broja (n,k), dobijamo:

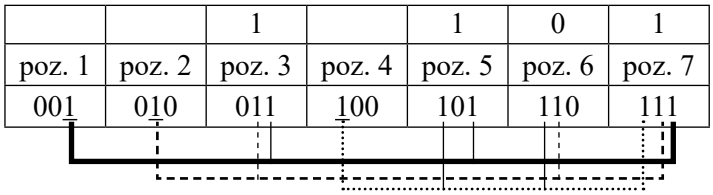
$$2^{n-k} - 1 = n$$

$$n = \log_2(n+1) + k.$$

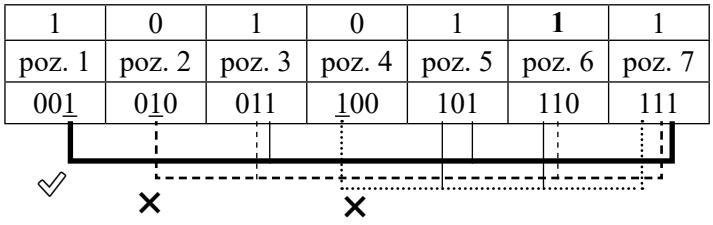
Dobijena je nelinearna relacija koja povezuje broj informacionih bita s ukupnim brojem bita u riječi. Međutim, činjenica da se u relaciji pojavljuje $\log_2(n+1)$ forsira da se n može prikazati kao $n = 2^m - 1$. Tako za $m = 3$, slijedi $n = 7$ i $k = 7 - m = 4$. Dakle, u tom slučaju dobijamo kod (7,4), dok, na primjer, za $m = 4$, slijedi $n = 15$ i $k = 15 - 4 = 11$, odnosno kod (15,11). Naredni Hemingovi kodovi su (31,26), (63,57) itd.

Sada ćemo na jednom intuitivnom primjeru uvesti Hemingov kod (7,4) i demonstrirati način rada ovog koda da bismo tokom daljeg izlaganja, u ovom i narednom poglavlju, ublažili pojedina ograničenja koja sada uvodimo, te ćemo dati matematički znatno precizniju formulaciju koda.

Posmatrajmo sada niz od četiri informaciona bita ($k = 4$) 1101. Formiraćemo kodnu riječ dužine $n = 7$, sa tri kontrolna bita ($m = n - k = 3$) postavljena na **pravilnim binarnim pozicijama**. Pravilne binarne pozicije su one koje se mogu zapisati kao 2^l, $l = 0, 1, \ldots, m-1$ (u predmetnom primjeru to su pozicije $2^0 = 1$, $2^1 = 2$ i $2^2 = 4$). Pozicije bita u kodnoj riječi zapisujemo i dekadno i, što je u ovom slučaju važnije, sa trobitnim binarnim zapisom. Na preostalim pozicijama upisujemo informacione bite (Slika V.5).



Slika V.5. Ilustracija Hemingovog (7,4) koda sa bitima parnosti na pravilnim binarnim pozicijama



Slika V.6. Dekodiranje greške u kodnoj riječi

Sa Slike V.5 uočavamo da su pozicije kontrolnih bita one na kojima se u binarnom zapisu samo na jednoj poziciji javlja jedinica. Bit parnosti sa posmatrane pozicije kontroliše one bite u kojima se u binarnom zapisu pozicije jedinica pojavljuje na istom mjestu kao kod bita parnosti. Prvi bit parnosti provjerava prvu, treću, petu i sedmu poziciju (neparne pozicije), drugi bit parnosti provjerava drugu, treću, šestu i sedmu poziciju, dok bit parnosti na poziciji četiri provjerava posljednja četiri bita u kodnoj riječi. Pod provjerom podrazumijevamo da je broj jedinica u posmatranom dijelu kodne riječi paran. Na osnovu ovakvog pravila, sada možemo da odredimo bite parnosti kao:

1 0 1 0 1 0 1.

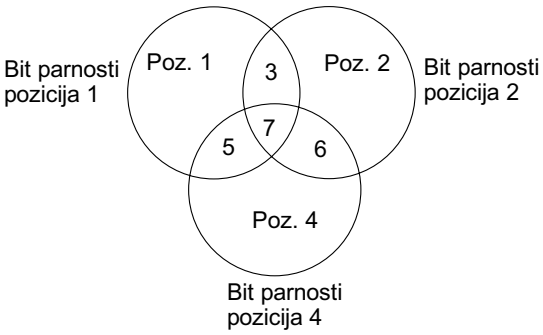
Ako nema greške u kodnoj riječi, onda će sve provjere parnosti biti zadovoljene.

Pretpostavimo da se greška dogodila na poziciji broj 6. U tom slučaju imamo situaciju koja je demonstrirana na Slici V.6. Vidimo da kontrole parnosti donose sljedeće odluke:

Parnost na poziciji 1:	1	1	1	1	✓
Parnost na poziciji 2:	0	1	1	1	✗
Parnost na poziciji 4:	0	1	1	1	✗.

Pozicija na kojoj se pogreška dogodila određuje se krajnje jednostavno, sabirajući pozicije na kojima imamo indikaciju pogreške, odnosno u ovom slučaju $2+4=6$ daje da se pogreška dogodila na šestoj poziciji! Predmetne provjere parnosti ukazuju na **sindrom**, odnosno poziciju na kojoj je kod „bolestan“.

Da bismo ilustrovali efektivnost predmetnog postupka, na Slici V.7 prikazan je Venov dijagram (u obliku skupovne algebre), koji pokazuje koji biti parnosti indiciraju koje greške. Sa dijagrama se vidi da ako samo jedan bit parnosti ukazuje na



Slika V.7. Ilustracija Hemingovog koda (7,4) sa provjerama parnosti na pravilnim binarnim pozicijama

grešku, to je greška koja se dogodila na tom bitu. Ako dva bita ukazuju na greške, to su situacije kada se greške događaju na trećem bitu kodne riječi (parnosti na pozicijama 1 i 2 nisu zadovoljene), petom (parnosti sa pozicija 1 i 4 nisu zadovoljene) i šestom (parnosti na pozicijama 2 i 4 nisu zadovoljene). Kada sve tri provjere parnosti ukazuju na grešku, to znači da se greška dogodila na poziciji $1+2+4=7$.

Na sličan način se može generisati Hemingov (15,11) kod s bitovima parnosti raspoređenim na pravilnim binarnim pozicijama (u ovom slučaju, to su pozicije 1, 2, 4 i 8).

Primjer V.4. Data je povorka od 11 bita: 10111010011. Izvršiti kodiranje Hemingovim kodom sa bitovima parnosti na pravilnim binarnim pozicijama.

Rješenje: Slika V.8 prikazuje način formiranja informacionih bita u ovom slučaju. Kontrolni bit na poziciji 1 (kontrolne bite smo radi lakše vidljivosti prikazali boldirano) kontroliše sve neparne pozicije. Kontrolni bit na poziciji 2 daje paran broj jedinica na pozicijama 2, 3, 6, 7, 10, 11, 14, 15. Kontrolni bit na poziciji 4 daje paran broj jedinica na pozicijama 4, 5, 6, 7, 12, 13, 14 i 15. Posljednji kontrolni bit na poziciji 8 provjerava posljednjih osam bita (pozicije od 8 do 15).

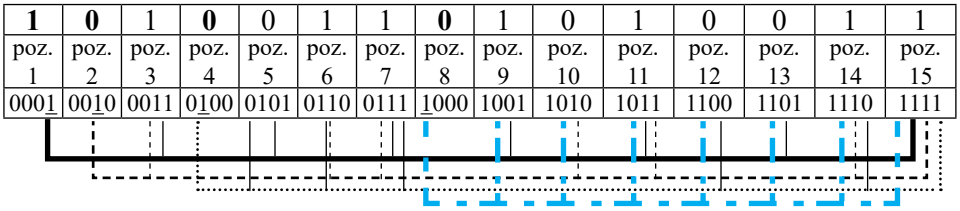
Blok kodovi rade na principu podjele ulaznog toka podataka u blokove od po k bita, koji se zatim kodiraju sa n bita, dodavanjem svakom bloku m redundantnih bita. Kod Hemingovog koda vidjeli smo da je pogodno da važi $m=\log_2(n+1)$. Međutim, postavlja se pitanje: šta ako se tok podataka ne dijeli u blokove ovako pogodne dužine? Na primjer, šta uraditi ako želimo da održimo funkcionalnost Hemingovog kodiranja? Pretpostavimo da imamo blokove dužine $k=8$. Očigledno su nam potrebna četiri bita parnosti na pozicijama 1, 2, 4 i 8 (pravilnim binarnim pozicijama). Ostale pozicije postavljamo do popunjavanja broja informacionih bita (pogledati Sliku V.9, gdje kodiramo 10110111), a one bite kojih nema ne uzimamo u obzir prilikom kodiranja, odnosno formiranja provjera parnosti.

Postoji još jedna dodatna mogućnost kod Hemingovih kodova, odnosno njihovo proširenje dodavanjem još jednog dodatnog bita parnosti koji provjerava čitavu riječ. Tako se dobija **prošireni Hemingov kod**. Iz koda (7,4) dobija se kod (8,4), iz (15,11) dobija se (16,11). Ovaj kod je u stanju da ispravi jednu pogrešku i da detektuje pojavu dvije pogreške. Uzmimo primjer koda sa četiri informaciona bita 1010. Postavimo tri bita parnosti na pravilne binarne pozicije, a dodatni bit na osmu poziciju:

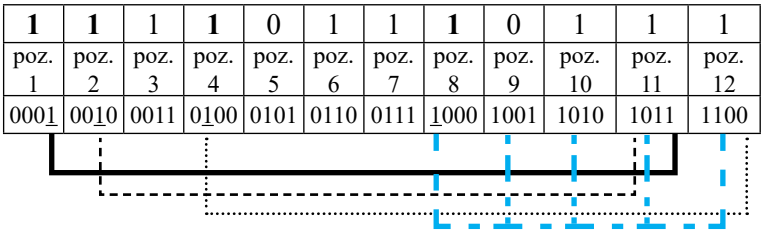
1 0 1 1 0 1 0 0.

U slučaju da se dogodi jedna pogreška, na primjer na petoj poziciji, imamo primljenu poruku:

1 0 1 1 **1** 1 0 0.



Slika V.8. Ilustracija Hemingovog (15,11) koda sa bitima parnosti na pravilnim binarnim pozicijama



Slika V.9. Skraćeni Hemingov kod

Provjerama parnosti dobijamo:

Pozicija 1: $1+1+1+0=1$ ✗ Pozicija 2: $0+1+1+0=0$ ✓

Pozicija 4: $1+1+1+0=1$ ✗ Pozicija 8: $1+0+1+1+1+1+0+0=1$ ✗.

Dakle, greška se dogodila na poziciji $1+4=5$, dok provjera parnosti na posljednjoj riječi samo kaže da se dogodila jedna pogreška.

Pogledajmo sada situaciju kada se dogode dvije pogreške. Neka se te greške dogode na pozicijama 1 i 6:

0 0 1 1 0 **0** 0 0.

Provjerama parnosti dobijamo:

Pozicija 1: $0+1+0+0=1$ ✗ Pozicija 2: $0+1+0+0=1$ ✗

Pozicija 4: $1+0+0+0=1$ ✗ Pozicija 8: $0+0+1+1+0+0+0+0=0$ ✓.

Dakle, sindromi na prva tri bita parnosti ukazuju da se greška dogodila (ovdje pokazuju na potpuno pogrešni bit, na poziciji $1+2+4=7$), ali bit na poziciji 8 ukazuje da greške u kodnoj riječi nema. Ovakva situacija je moguća samo kada su se dogodile dvije pogreške u kodnoj riječi (ili paran broj pogreški).

Primjer V.5. Vjerovatnoća greške na jednom bitu je p, a greške su i.i.d. procesi. Posmatrati slučajeve Hemingovih kodova (7,4) i (8,4). Odrediti vjerovatnoće uspješnog prenosa (uključujući ispravku greške), vjerovatnoću da se greške detektuju, te vjerovatnoću da se greška ne može detektovati.

Rješenje: Vjerovatnoća da prvi kod primi poruku koju može uspješno dekodirati jednaka je:

$$Q = (1-p)^7 + 7p(1-p)^6 = (1+6p)(1-p)^6,$$

dok je vjerovatnoća greške u dekodiranju jednaka (slučaj kada se desi više od jedne pogreške):

$$\mathrm{P}_E = 1 - Q.$$

Kod drugog koda vjerovatnoća da kod radi, odnosno da je u stanju da ispravi pogrešku, jednaka je:

$$Q' = (1-p)^8 + 8p(1-p)^6 = (1+7p)(1-p)^7.$$

Odnos vjerovatnoće uspješnog dekodiranja za ova dva koda je:

$$\frac{Q'}{Q} = \frac{(1+7p)(1-p)}{1+6p}.$$

Vjerovatnoća da kod detektuje pogreške (to se dešava u slučaju svih parnih brojeva pogreški osim za svih osam pogreški) je:

$$Q_d = 28p^2(1-p)^6 + 70p^4(1-p)^4 + 28p^6(1-p)^2.$$

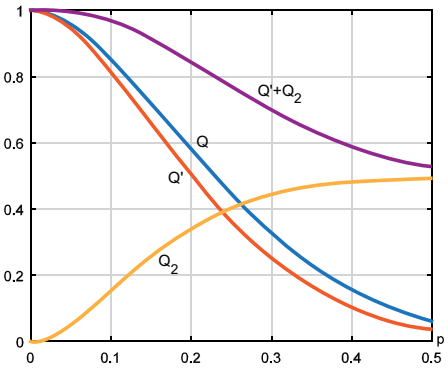
Predmetni kod nije u stanju da ispravi, odnosno detektuje pogrešku, sa vjerovatnoćom:

$$\mathrm{P}'_E = 1 - Q' - Q_d.$$

Na Slici V.10 prikazane su vjerovatnoće uspješnog dekodiranja kod Hemingovih kodova (7,4) (plava linija) i (8,4) (crvena linija). Očigledno je da je veća vjerovatnoća uspješnog dekodiranja (ispravnog prenosa i ispravke jedne pogreške) kod kraćeg koda nego kod dužeg jer se u dužoj kodnoj riječi vjerovatnije generišu pogreške. Vjerovatnoća da kod (8,4) detektuje pogrešku prikazana je žutom linijom. Ukupna vjerovatnoća pravilnog dekodiranja i detekcije pogreški (ljubičasta linija) veća je nego vjerovatnoća pravilnog dekodiranja kod koda (7,4).

V.4 Hemingova distanca, Hemingova težina i pakovanje sfera

Hemingova distanca je mjera razlike između dvije kodne riječi. Označava se kao $d_H(\mathbf{c}_1,\mathbf{c}_2)$, gdje su $\mathbf{c}_i$, $i = 1,2$ kodne riječi iste dužine, prikazane putem vektora. Hemingova distanca je broj pozicija na kojima se dvije kodne riječi razlikuju. Na primjer, kodne riječi:



Slika V.10. Vjerovatnoće kod Hemingovog (7,4) i (8,4) koda: plava linija – vjerovatnoća pravilnog dekodiranja (7,4) koda; crvena linija – vjerovatnoća pravilnog dekodiranja (8,4) koda; žuta linija – vjerovatnoća detekcije pogreške kod (8,4) koda; ljubičasta linija – ukupna vjerovatnoća pravilnog dekodiranja i detekcije pogreške kod (8,4) koda

$$\begin{matrix} 1 \\ 1 \end{matrix} \quad \begin{matrix} 0 \\ 0 \end{matrix} \quad \boxed{\begin{matrix} 1 \\ 0 \end{matrix}} \quad \begin{matrix} 1 \\ 1 \end{matrix} \quad \boxed{\begin{matrix} 1 \\ 0 \end{matrix}} \quad \boxed{\begin{matrix} 0 \\ 1 \end{matrix}} \quad \boxed{\begin{matrix} 1 \\ 0 \end{matrix}}$$

razlikuju se na četiri pozicije, pa je Hemingovo rastojanje među ovim riječima 4. Hemingova težina je broj jedinica u jednoj kodnoj riječi. Označava se kao $w_H(\mathbf{c})$. U prethodnom primjeru prva kodna riječ ima težinu 5, dok druga ima težinu 3. Između ove dvije funkcije (distance i težine), može se uspostaviti sljedeća relacija:

$$d_H(\mathbf{c_1}, \mathbf{c}_2) = w_H(\mathbf{c}_1 \oplus \mathbf{c}_2),$$

gdje $\mathbf{c}_1 \oplus \mathbf{c}_2$ označava operaciju ekskluzivno ili primijenjenu bit po bit. Za kod se definiše minimalno Hemingovo rastojanje kao minimalna distanca između dvije različite kodne riječi koda:

$$\min_{\substack{\mathbf{c}_1, \mathbf{c}_2 \\ \mathbf{c}_1 \neq \mathbf{c}_2}} d_H(\mathbf{c}_1, \mathbf{c}_2).$$

Minimalno Hemingovo rastojanje je fundamentalna granica za kodove. Posmatrajmo jednostavni binarni kod sa dva bita {00, 01, 10, 11}. Vidimo da je minimalno rastojanje među bilo koje dvije različite kodne riječi koda jednako 1. Dakle, da bi kod bio jednoznačno dekodabilan, mora da ima minimalno Hemingovo rastojanje 1, odnosno sve kodne riječi moraju biti različite. Posmatrajmo sada slučaj da na dva bita dodajemo bit parnosti. Znamo da bi ovakva struktura trebala da omogući detekciju jedne pogreške {000, 011, 101, 110}. Različite kodne riječi se razlikuju na dvije pozicije. Dakle, da bi kod mogao da detektuje jednu pogrešku, minimalno Hemingovo rastojanje među kodnim riječima mora da bude 2. Idemo korak dalje: kod sa majoritetnom logikom sa tri bita {000, 111} u stanju je da ispravno dekodira situaciju jer ima minimalno Hemingovo rastojanje 3. Slično, kod sa ponavljanjem

četiri bita {0000, 1111} u stanju je da ispravi jednu pogrešku, ali može da detektuje dvije pogreške (kada primi dvije jedinice i nule konstatuje da su se u kanalu dogodile dvije pogreške, ali ne može da ih ispravi). Ovaj kod ima minimalno Hemingovo rastojanje koje je jednako četiri. Možemo ići još jedan korak dalje: ako imamo kod sa ponavljanjem i pet bita {00000, 11111}, u stanju je da ispravi dvije pogreške. Premda nismo dali eksplicitan dokaz, predmetno racionalisanje nas vodi na zaključak da ako je kod u stanju da ispravi w pogreški i da detektuje dodatnih e pogreški, njegovo minimalno Hemingovo rastojanje mora da bude:

$$d = 2w + e + 1.$$

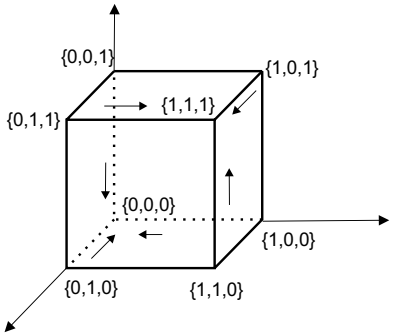
Imamo rijetku priliku da ovu situaciju možemo lijepo da vizuelizujemo kod repetitivnog koda sa tri bita (Slika V.11). Korektne kodne riječi prikazujemo tjemenima kocke {000, 111}. Situacije kada se jedan bit promijeni pridružujemo, po Hemingovom rastojanju, najbližoj ispravnoj kodnoj riječi. Sada vidimo i zbog čega je neophodno da minimalno Hemingovo rastojanje bude 3. Naime, sve riječi koje se na jednom mjestu razlikuju od korektne kodne riječi pridružujemo toj kodnoj riječi (one sa dvije nule pridružujemo riječi 000, a one sa dvije jedinice pridružujemo 111). Ovo se u terminologiji teorije informacija i kodova naziva **sferom** radijusa 1. Druga ispravna kodna riječ je takođe okružena sferom radijusa 1. Ove sfere se ne smiju ni dodirivati ni sjeći što dalje znači da između njih mora postojati dodatna razlika na jednom mjestu. Dakle, ukupno mora postojati minimalno Hemingovo rastojanje 3 (radijus jedne sfere + radijus druge sfere + prostor između sfera) da bi kod ispravljao jednu pogrešku.

Iz predmetnog razmatranja slijedi zaključak vezan za dimenzije koda. Ako imamo k informacionih bita, sa njima generišemo 2^k ispravnih kodnih riječi. Ove riječi su prikazane putem n bita u kodnoj riječi. Dakle, u uslovima grešaka u prenosu, mi možemo primiti bilo koju od 2^n riječi koje se mogu konstruisati sa n bita. Svakoj ispravnoj kodnoj riječi moramo da pridružimo barem još n riječi koje se od nje razlikuju na jednom mjestu, čineći sferu **zapremine** $1+n$ (u terminologiji teorije kodova broj riječi u nekoj sferi naziva se zapremina). Dakle, prostor 2^n mogućih stanja mora da bude izdijeljen na sfere zapremine $1+n$ tako da u prostoru stane makar 2^k nedodirujućih i nepreklapajućih sfera. Odnosno, mora da važi:

$$\frac{2^n}{1+n} \geq 2^k.$$

Kada malo bolje sagledamo, ova relacija je kod Hemingovog koda zadovoljena jednakošću. Najbolje spakovani sistem za ispravljanje jedne pogreške dobija se kada je:

$$2^n = 2^k(1+n) \Rightarrow n = k + \log_2(1+n).$$



Slika V.11. Ilustracija kodova sa minimalnim Hemingovim rastojanjem $d = 3$

Predmetna granica koja se može uspostaviti između kodnih riječi naziva se granicom pakovanja sfera. Može se proširiti i na slučaj kada imamo više pogreški. Na primjer, ako imamo dvije pogreške, treba nam minimalno Hemingovo rastojanje 5. Naime, oko svake ispravne kodne riječi konstruišemo sferu radijusa 2 (pored korektno prenesene riječi, obuhvata i one riječi koje su na rastojanju 1 i 2 od ove riječi), te se dvije sfere moraju razmaknuti još minimalno za jednu poziciju. Zapremina sfere oko korektne pozicije u ovom slučaju je:

$$1 + n + \binom{n}{2}$$

tako da važi:

$$\frac{2^n}{1 + n + \binom{n}{2}} \geq 2^k \Rightarrow 2^n \geq 2^k \left[1 + n + \binom{n}{2} \right].$$

Hemingova distanca mora zadovoljavati sljedeće osobine da bi se zvala distancom – rastojanjem ili metrikom:

1. nenegativnost $d_H(\mathbf{c_1}, \mathbf{c}_2) \geq 0$;
2. nulto rastojanje se postiže samo za identične argumente, odnosno $d_H(\mathbf{c_1}, \mathbf{c}_2) = 0$ je ekvivalentno sa $\mathbf{c_1} = \mathbf{c}_2$;
3. simetričnost $d_H(\mathbf{c}_1, \mathbf{c}_2) = d_H(\mathbf{c}_2, \mathbf{c}_1)$;
4. nejednakost trougla $d_H(\mathbf{c}_1, \mathbf{c}_2) + d_H(\mathbf{c}_2, \mathbf{c}_3) \geq d_H(\mathbf{c}_1, \mathbf{c}_3)$.

Ove osobine se za Hemingovu distancu dokazuju tako što se dokažu da važe za svaki pojedinačni bit, pa ako je zadovoljeno, onda važi i za čitavu riječ. Imajmo na umu da se distanca nad riječju dobija standardnim aritmetičkim operacijama (konkretno, sabiranjem prirodnih brojeva) nad distancama nad bitovima.

$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$d_H(\mathbf{c}_1,\mathbf{c}_2)$	$d_H(\mathbf{c}_2,\mathbf{c}_3)$	$d_H(\mathbf{c}_1,\mathbf{c}_2)+d_H(\mathbf{c}_2,\mathbf{c}_3)$	$d_H(\mathbf{c}_1,\mathbf{c}_3)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	2	0
0	1	1	1	0	1	1
1	0	0	1	0	1	1
1	0	1	1	1	2	0
1	1	0	0	1	1	1
1	1	1	0	0	0	0

Tabela V.1. Dokaz da je Hemingovo rastojanje ispravna metrika

Prva osobina kod bita je, očigledno, tačna jer je $d_H(0,0)=0\oplus 0=0$, $d_H(0,1)=0\oplus 1=1$, $d_H(1,0)=1\oplus 0=1$ i $d_H(1,1)=1\oplus 1=0$. Iz prethodnog vidimo da smo dobili nulu samo za iste vrijednosti argumenata. Vidimo da slično važi i simetričnost, čime smo jednim potezom dokazali tri stava (kako se to ponekada kaže: jednim metkom ubili tri zeca). Preostaje da se dokaže posljednji stav, što je urađeno u Tabeli V.1. Poređenjem pretposljednje i posljednje kolone jasno slijedi da je zadovoljena nejednakost trougla, čime je Hemingovo rastojanje ispunilo sve potrebne i dovoljne osobine da bi se nazvalo metrikom.

V.5 Zadaci i softverska realizacija

V.5.1 Riješeni zadaci

5.1. Kodirati podatke o muškoj djeci rođenoj 12. 02. 2013. u Podgorici (21), putem JMBG. Ukupno ih je troje toga dana rođenih (od 000 do 002).

Rješenje: Sva djeca imaju početak JMBG 120201321 i razlikuju se na naredne tri cifre sa 000, 001 i 002. Sračunajmo u sva tri slučaja težinske sume da bismo odredili odgovarajuće kontrolne cifre:

$$7\cdot 1+6\cdot 2+5\cdot 0+4\cdot 2+3\cdot 0+2\cdot 1+7\cdot 3+6\cdot 2+5\cdot 1+4\cdot 0+3\cdot 0+2\cdot 0=67$$
$$7\cdot 1+6\cdot 2+5\cdot 0+4\cdot 2+3\cdot 0+2\cdot 1+7\cdot 3+6\cdot 2+5\cdot 1+4\cdot 0+3\cdot 0+2\cdot 1=69$$
$$7\cdot 1+6\cdot 2+5\cdot 0+4\cdot 2+3\cdot 0+2\cdot 1+7\cdot 3+6\cdot 2+5\cdot 1+4\cdot 0+3\cdot 0+2\cdot 2=71.$$

Dodavanjem posljednje cifre treba da bude zadovoljena djeljivost sa 11:

$$(67+d)\%11=(1+d)\%11=0\Rightarrow d=10\Rightarrow d=0$$
$$(69+d)\%11=(3+d)\%11=0\Rightarrow d=8$$
$$(71+d)\%11=(5+d)\%11=0\Rightarrow d=6.$$

U relacijama smo koristili osobinu **kongruencije**, odnosno da je

$$(a+b)\,\%\,p=((a\,\%\,p)+(b\,\%\,p))\,\%\,p.$$

Dakle, ispravni JMBG su za navedenu djecu:

1202013210000

1202013210018

1202013210026.

5.2. Odrediti kontrolnu cifru za ISBN-13 kod 978-3-16-148410.

Rješenje: Odredimo težinsku sumu prvih 12 cifara:

$$1\cdot 9+3\cdot 7+1\cdot 8+3\cdot 3+1\cdot 1+3\cdot 6+1\cdot 1+3\cdot 4+1\cdot 8+3\cdot 4+1\cdot 1+3\cdot 0=100\,.$$

Dakle, posljednja (kontrolna) cifra, da se zadrži djeljivost sa 10, treba da bude 0, pa je ispravan ISBN-13 kod:

978-3-16-148410-0.

5.3. U praksi postoji značajan broj ARQ (engl. *Automatic Repaet Request*) kodova kod kojih dekoder može da prepozna da se greška u poruci dogodila i da automatski generiše zahtjev za ponavljanje poruke. Jedan od tih kodova naziva se kod „3 od 7“ i kod njega u poruci uvijek postoje 3 jedinice (ili nule, u zavisnosti od konvencije) i ostatak nula. Dekoder broji nule i jedinice i ako naiđe na neregularnost, traži ponovno slanje poruke. Koliki je kodni odnos ovog koda? Kolika je vjerovatnoća da se dogodila pogreška i da je dekoder zahtijevao ponovno slanje poruke? Kolika je vjerovatnoća da dekoder za pogrešnu poruku utvrdi da je ispravna? Pretpostaviti da je vjerovatnoća pogreške p i da su pogreške na pojedinim bitovima nezavisne. Ponoviti prethodnu proceduru ako je $p=0.1$ i ako je vjerovatnoća da se greška dogodila na narednom bitu, pod uslovom da se dogodila na prethodnom, jednaka 0.3. Pretpostaviti da je poruka od 7 bita nezavisna od svih ostalih poruka u sistemu.

Rješenje: Već smo vidjeli da je kodni odnos ovog koda:

$$R=\frac{\log_2 35}{7}\approx 0.7328$$

jer se sa 7 bita može generisati

$$\binom{7}{3}=35$$

poruka sa 3 jedinice. U slučaju i.i.d. grešaka, u kodnoj riječi nema pogreški sa vjerovatnoćom:

$$(1-p)^7.$$

Neparan broj pogreški koji se uvijek može detektovati i zahtijevati ponovno slanje poruke dešava se s vjerovatnoćom:

$$7p(1-p)^6+\binom{7}{3}p^3(1-p)^4+\binom{7}{5}p^5(1-p)^2+\binom{7}{7}p^7=$$
$$=7p(1-p)^6+35p^3(1-p)^4+21p^5(1-p)^2+p^7.$$

Kod dvije pogreške postoji $\binom{7}{2}=21$ kombinacija, od kojih se mogu detektovati one kod kojih se 2 jedinice promijene, odnosno 2 nule. Takvih kombinacija je ukupno: $\binom{3}{2}+\binom{4}{2}=9$. Kod četiri greške (kombinacija je 35) možemo da detektujemo situacije kada se sve četiri nule promijene (1 kombinacija), 3 nule i jedna jedinica (12 kombinacija) i jedna nula i tri jednice (4 kombinacije). Ukupan broj kombinacija, kada detektujemo grešku, jeste 17 za četiri pogreške. Konačno, kod 6 pogreški povoljne su situacije kada imamo izmjene četiri nule i dvije jedinice, a takvih kombinacija je 3. Dakle, u slučaju i.i.d. procesa, ukupna vjerovatnoća da možemo detektovati pogrešku je:

$$P_{\text{det}}=7p(1-p)^6+9p^2(1-p)^5+35p^3(1-p)^4$$
$$+17p^4(1-p)^3+21p^5(1-p)^2+3p^6(1-p)+p^7.$$

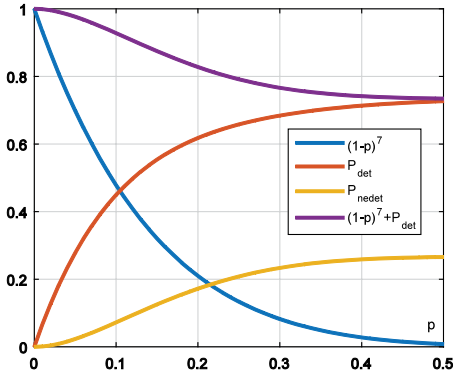
Preostala vjerovatnoća predstavlja situaciju kada kod ne odgovara svojoj svrsi, odnosno kada nije u stanju da detektuje pogreške:

$$P_{\text{nedet}}=1-(1-p)^7-P_{\text{det}}=6p^2(1-p)^5+18p^4(1-p)^3+4p^6(1-p).$$

Na Slici V.12 prikazane su vjerovatnoće prenosa bez pogreške (plava linija – $(1-p)^7$), vjerovatnoća detekcije pogreške (crvena linija – P_{det}), vjerovatnoća nedetekcije (žuta linija – P_{nedet}) i ukupna vjerovatnoća kada kod služi svrsi, odnosno prenosi ispravnu poruku ili detektuje pogrešnu (ljubičasta linija – $(1-p)^7+P_{\text{det}}$).

Izvođenje za slučaj kada imamo greške koje su međusobno zavisne znatno je složenije. Označimo sa p vjerovatnoću greške na bitu kada prethodno nije bilo pogreške. Neka je r vjerovatnoća greške kada bitu prethodi pogrešni simbol. Vjerovatnoća da nema grešaka u sistemu je ponovo kao i kod i.i.d. procesa: $(1-p)^7$.

Vjerovatnoća jedne pogreške je jednaka vjerovatnoći da se greška dogodi na svakom pojedinačnom bitu:



Slika V.12. Kod „3 od 7“ – vjerovatnoće: uspješnog prenosa, detekcije pogreške, nedetekcije pogreške, zbir detektovane greške i uspješnog prenosa

$$P_1 = p(1-r)(1-p)^5 + (1-p)pr(1-p)^4 + \cdots + (1-p)^5 p(1-r) + (1-p)^6 p =$$
$$= 6p(1-r)(1-p)^5 + p(1-p)^6 = p(1-p)^5[6-r+1-p] = p(1-p)[7-r-p].$$

Vjerovatnoća dvije pogreške je:

$$P_2 = 6pr(1-r)(1-p)^4 + 10p^2(1-r)^2(1-p)^3 + 5p^2(1-r)(1-p)^4,$$

dok je sada vjerovatnoća pojave tri pogreške:

$$P_3 = 4pr^2(1-p)^3(1-r) + 12p^2r(1-p)^2(1-r)^2 + 8p^2r(1-p)^3(1-r)$$
$$+ 4p^3(1-p)(1-r)^3 + 6p^3(1-p)^2(1-r)^2 + pr^2(1-p)^4.$$

Ostale vjerovatnoće su jednake (za pojavu 4, 5, 6 i 7 pogreški):

$$P_4 = 3pr^3(1-r)(1-p)^2 + 9p^2r^2(1-r)^2(1-p) + 9p^2r^2(1-r)(1-p)^2 +$$
$$+ 3p^3r(1-r)^3 + 9p^3r(1-r)^2(1-p) + p^4(1-r)^3 + pr^3(1-p)^3$$
$$P_5 = 2pr^4(1-r)(1-p) + 5p^2r^3(1-r)^2 +$$
$$+ 8p^2r^3(1-r)(1-p) + 5p^3r^2(1-r)^2 + pr^4(1-p)^2$$
$$P_6 = pr^5(1-r) + 5p^2r^4(1-r) + pr^5(1-p)$$
$$P_7 = pr^6.$$

Svaki neparni broj pogreški je detektabilan, što se ne može reći za paran broj pogreški. Da bismo odredili tačnu vjerovatnoću greške za paran broj pogreški, morali bismo da poznajemo tačnu kodnu knjigu, ali dobru procjenu možemo da odredimo i na drugi način. Vidjeli smo da kada postoje dvije pogreške od 21 moguće varijante, 9 je detektabilnih, pa možemo zaključiti da u $9/21 = 4/7$ situacija sa dvije pogreške možemo da detektujemo grešku. Kod četiri pogreške u stanju smo

da detektujemo greške u 17 od 35 situacija, dok smo kod 6 pogreški u stanju da detektujemo greške u 3 od 7 situacija. Dakle, vjerovatnoća detekcije pogreške je:

$$P_{\text{det}} = P_1 + \frac{3}{7}P_2 + P_3 + \frac{17}{35}P_4 + P_5 + \frac{3}{7}P_6 + P_7.$$

Vjerovatnoća da se greška ne može detektovati je:

$$P_{\text{ne det}} = \frac{4}{7}P_2 + \frac{18}{35}P_4 + \frac{4}{7}P_6.$$

U slučaju sa Slike V.12 za $p = 0.1$, kada su pogreške po bitovima međusobno nezavisne, vjerovatnoća uspješnog prenosa je približno 0.48, dok je vjerovatnoća detektovane pogreške 0.43. Sljedstveno, vjerovatnoća nedetektovane pogreške je aproksimativno 0.07. U slučaju kada imamo korelaciju između pogreški za $p = 0.1$ i $r = 0.3$, vjerovatnoća uspješnog prenosa ostaje ista, ali raste vjerovatnoća nedetektovane pogreške na $P_{ne\,\text{det}} \approx 0.1653$, čime je i na ovom primjeru ukazana činjenica da kodovi rade lošije u uslovima koreliranih – povezanih grešaka.

5.4. Često se u vojnim komunikacijama, kao i u nekim drugim aplikacijama, koristi sistem prenosa podataka većinom glasova. Isti simbol (nula ili jedan) šalje se neparan broj puta i odluka koji je simbol primljen donosi se na osnovu onog simbola koji je unutar jednog signalizacionog intervala primljen veći broj puta. Neka je vjerovatnoća greške na jednom bitu 0.1. Kolika je greška u dekodiranju poruke ako se svaki informacioni bit ponavlja 3, odnosno 5 puta? Kolike su vjerovatnoće greške kada je pojedinačni bit primljen pogrešno sa vjerovatnoćom 0.3?

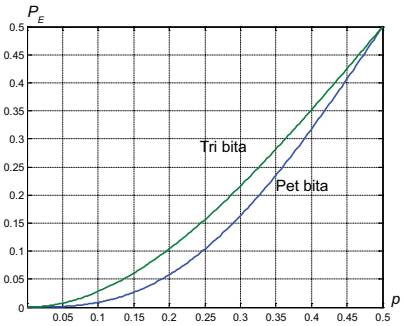
Rješenje: Ako imamo poruku od tri bita, grešku ćemo detektovati kada dva bita primimo sa greškom, odnosno kada primimo grešku na sva tri bita:

$$P_E = 3p^2(1-p) + p^3.$$

Broj 3, koji množi prvi član izraza, podrazumijeva da bi ispravan bit mogao biti na bilo kojoj od tri pozicije u kodu. Dakle, vjerovatnoća greške za $p = 0.1$ jednaka je $P_E = 0.028$, dok je za $p = 0.3$ $P_E = 0.216$. Kod majoritetne logike sa pet bita, greška u dekodiranju se dešava kod grešaka na 3 ili više bita. Vjerovatnoća greške onda iznosi:

$$P_E = 10p^3(1-p)^2 + 5p^4(1-p) + p^5.$$

Za vjerovatnoću greške $p = 0.1$, vjerovatnoća P_E jednaka je $P_E = 0.0086$, dok je za $p = 0.3$ vjerovatnoća nedetekcije pogreške $P_E = 0.1631$. Na Slici V.13 prikazana je vjerovatnoća greške za slučaj kada imamo većinu od tri bita (plava linija) i za slučaj kada imamo većinu od 5 bita (zelena linija). Jasno se bolji rezultati postižu u drugom slučaju, ali se to plaća manjim kodnim odnosom $R = 1/5$ u odnosu na $R = 1/3$.



Slika V.13. Vjerovatnoća greške kod kodova sa majoritetnom logikom od tri bita (zelena linija) i pet bita (plava linija) u zavisnosti od vjerovatnoće greške po bitu p

5.5. Kombinaciju od $k_1 \times k_2$ bita treba kodirati pravougaonim kodom. Ako je vjerovatnoća greške na bilo kom bitu jednaka p i ako su pogreške na bitima međusobno nezavisne, odrediti vjerovatnoće da: (a) nema greške u sistemu; (b) kod detektuje i ispravi pogrešku; (c) kod detektuje, ali ne može da ispravi pogrešku; (d) kod pogrešno ispravi nastalu pogrešku.

Rješenje: (a) Vjerovatnoća da nema greške u sistemu jednaka je $(1-p)^{(k_1+1)(k_2+1)}$, jer je $(k_1+1)(k_2+1)$ ukupan broj bita koji su kodirani u ovoj poruci.

(b) Vjerovatnoća da se dogodi pogreška i da je možemo ispraviti jednaka je:

$$(k_1+1)(k_2+1)p(1-p)^{(k_1+1)(k_2+1)-1}.$$

(c) Kod će biti u stanju da detektuje pojavu pogreške ako se to dogodi na dva bita. Naime, ako se pogreška dogodi u jednoj vrsti i/ili koloni, onda će ta vrsta biti korektna sa stanovišta bita parnosti, ali će se te greške pojaviti u kolonama/vrstama, što će ukazati na dvije pogreške. U slučaju da se greške dogode u različitim kolonama/ vrstama, onda će se na više od dva bita parnosti indicirati da je došlo do pogreški, što ponovo ne možemo ispraviti, ali možemo detektovati. Dakle, vjerovatnoća dvije pogreške kod pravougaonog koda je:

$$\frac{(k_1+1)(k_2+1)[(k_1+1)(k_2+1)-1]}{2}p^2(1-p)^{(k_1+1)(k_2+1)-2}.$$

Situacija sa više od tri pogreške znatno je komplikovanija i neće biti dalje analizirana, jer u tim slučajevima postoje obje varijante: i da se može detektovati pogreška, ali ne i ispraviti i varijanta da se konstatuje da se dogodila jedna pogreška, te da se ta pogreška ispravi iako se dogodilo više od jedne greške.

5.6. Poruku 110011 treba kodirati sa pravougaonim kodom (12,6). Kodna riječ je formirana red po red. Do greške u rezultujućoj poruci je došlo na poziciji 7. Izvršiti dekodiranje. Ponoviti postupak ako je do grešaka došlo na pozicijama 5 i 11.

1	1	0	0
0	1	1	0
1	0	1	0

(a)

1	1	0	0
0	1	**0**	0✕
1	0	1✕	0✕

(b)

1	1	0	0
1	1	1	0✕
1✕	0	**0**✕	0

(c)

Tabela V.2. Pravougaoni kod (12,6): (a) Bez pogreški; (b) Sa jednom pogreškom; (c) Sa dvije pogreške

Rješenje: Dakle, u ovom postupku kodiranja imamo da su biti raspoređeni kako je dato u Tabeli V.2(a). Kolonu i vrstu s kontrolnim bitima popunili smo tako da bude obezbijeđen paran broj bita. Tabela V.2(b) daje slučaj greške na 7. bitu (pogreška je na odgovarajući način vizuelizovana). Bitovi parnosti, koji provjeravaju drugu vrstu i treću kolonu, indiciraju pogrešku, kao i bit parnosti za čitavu tabelu. Dakle, znamo poziciju pogreške i u stanju smo da pravilno dekodiramo da je poslata poruka 110011. U narednom primjeru (Tabela V.2(c)) imamo pogreške na petom i jedanaestom (kontrolnom) bitu. Vidimo da su se greške dogodile u drugoj vrsti, prvoj i trećoj koloni, a bit parnosti za čitavu tabelu sugeriše da pogreške nema, odnosno da se dogodio paran broj pogreški. Premda u ovom slučaju kod ne može da ispravi grešku, ipak ima sposobnost da detektuje situaciju kada se dogodilo više od jedne pogreške.

5.7. Dekodirati poruku 1000010000101001 koja je kodirana pravougaonim kodom. Biti parnosti su raspoređeni na posljednjim pozicijama u kodnoj riječi.

Rješenje: Kod ima $n = 16$ bita, pa zaključujemo da može biti (16,9), koji je dat u Tabeli V.3. Uočimo da su prvih 9 bita informacioni, a da su ostalih 7 kontrolni. Iz tabele se vidi da se greška dogodila u prvoj vrsti i trećoj koloni, pa zaključujemo da treba izmijeniti bit koji je vizeulno naznačen, tako da je ispravljena poruka:

$$10\mathbf{\underline{1}}001000.$$

5.8. Dat je trougaoni kod (15,10). Izvršiti kodiranje poruke 1101101010. Poruka se formira red po red trougla. Dekodirati poruku ako je do greške došlo na poziciji 6 i ponoviti postupak ako je do grešaka došlo na pozicijama 3 i 7.

Rješenje: Tabela V.4 prikazuje predmetni trougaoni kod, situaciju kada postoji jedna pogreška i kada imamo dvije pogreške. Kodirana poruka (formirana red po red) je:

$$1101101101000$$

i prikazana je u Tabeli V.4(a). Kada imamo jednu pogrešku (Tabela V.4(b)), ispravno dekodiramo da se pogreška dogodila u prvoj koloni i na presjeku dijela koda kojeg kontroliše druga provjera parnosti (druga vrsta, četvrta kolona). U tom presjeku nalazi se bit na kome se dogodila pogreška, pa se poruka dekodira ispravno.

1	0	**0**	0✕
0	0	1	1
0	0	0	0
1	0	0✕	1✕

Tabela V.3. Pravougaoni kod (16,9) iz zadatka 5.7

1	1	0	1	1
1	0	1	1	
0	1	0		
0	0			
0				

(a)

1	1	0	1	1
0	0	1	1✕	
0	1	0		
0	0			
0✕				

(b)

1	1	**1**	1✳	1✕
1	0	**0**	1✕	
0	1	0		
0	0			
0				

(c)

Tabela V.4. Trougaoni kod (15,10): (a) Bez pogreški; (b) Sa jednom pogreškom; (c) Sa dvije pogreške 1010010001.

Primjer s dvije pogreške iz Tabele V.4(c) ima dva kontrolna bita koji indiciraju pogrešku. Stoga detektujemo da se pogreška dogodila na bitu koji je označen zvjezdicom. Međutim, promjena ovog bita donosi dodatnu (treću) pogrešku u dekodiranu kodnu riječ:

$$11\mathbf{10}10\mathbf{0}010.$$

5.9. Poruka je kodirana trougaonim kodom (15,10) i na prijemu glasi 101101000101100. Biti parnosti su dati na posljednjim mjestima u kodnoj riječi. Izvršiti dekodiranje primljene poruke.

Rješenje: Formirajmo Tabelu V.6, smještajući 10 prvih bita na pozicije informacionih bita i preostalih pet na pozicije kontrolnih bita. Prve dvije kontrole parnosti ukazuju na pogrešku, pa se greška detektuje na četvrtoj poziciji u prvom redu. Dekodirana je poruka:

$$1010010001.$$

1	0	1	**1**	0✕
0	1	0	1✕	
0	0	1		
1	0			
0				

Tabela V.5. Trougaoni kod (15,10) iz zadatka 5.9

5.10. Poruku 1011 kodirati Hemingovim kodom (7,4). Generisati pogrešku na poziciji 6, koju treba detektovati i ispraviti. Hemingov kod neka bude s pravilnim binarnim rasporedom bitova.

Rješenje: Formirajmo kod na Slici V.14(a). Poslata kodna riječ je:

$$0110011.$$

Nakon izmjene na šestom bitu, riječ je (Slika V.14(b)):

$$011000\underline{\mathbf{0}}1.$$

Sindromski biti (parnosti) na pozicijama 2 i 4 indiciraju pogrešku, pa konstatujemo da se greška dogodila na poziciji 6 i dekodiramo ispravno informacione bite.

5.11. Data su 4 informaciona bita 0101. Kodirajmo ih proširenim Hemingovim kodom sa bitovima parnosti na pravilnim binarnim pozicijama. Izvršiti korekciju pogreške ako se dogodila na petoj poziciji u kodnoj riječi, a zatim izvršiti detekciju pogreški ako su se dogodile na četvrtoj i petoj poziciji u kodnoj riječi.

Rješenje: Kodna riječ je:

$$c_1\ c_2\ 0\ c_3\ 1\ 0\ 1\ c_4.$$

Prvi bit parnosti kontroliše bite na pozicijama 1, 3, 5 i 7, pa mora biti $c_1 = 0$. Bit parnosti na poziciji 2 kontroliše bite na pozicijama 2, 3, 6 i 7, pa mora biti $c_2 = 1$. Bit parnosti na poziciji četiri kontroliše bite na pozicijama 4, 5, 6 i 7, pa mora biti $c_3 = 0$. Konačno, posljednji bit parnosti postavlja se tako da bude zadovoljena provjera parnosti za čitavu riječ, a u ovom slučaju to je $c_4 = 1$. Dakle, ispravna kodna riječ je:

$$0\ 1\ 0\ 0\ 1\ 0\ 1\ 0.$$

Ako se greška dogodila na petoj poziciji u kodnoj riječi, imamo primljenu poruku:

$$0\ 1\ 0\ 0\ \underline{\mathbf{0}}\ 0\ 1\ 0.$$

Provjere parnosti daju:

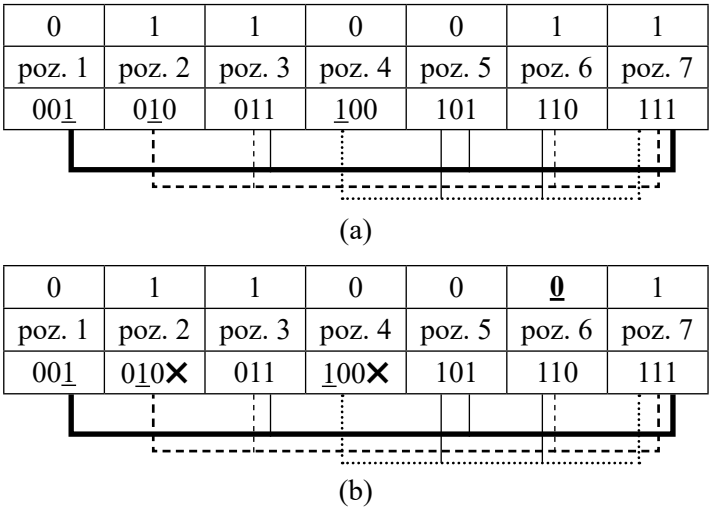
Pozicija 1: $0+0+0+1=1$ ✗ Pozicija 2: $0+1+0+1=0$ ✓

Pozicija 4: $0+0+0+1=1$ ✗ Pozicija 8: $0+1+0+0+0+0+1+1=1$ ✗.

Dakle, zaključujemo da se greška dogodila na poziciji $1+4=5$, a to potvrđuje i bit parnosti na posljednjoj poziciji (za čitavu riječ). Na ovaj način možemo ispraviti bit na poziciji 5:

$$0\ 1\ 0\ 0\ 1\ 0\ 1\ 0$$

i dekodirati poruku (uzimajući bite sa pozicija koje nisu pravilne binarne) 0101.



Slika V.14. Hemingov kod (7,4) iz zadatka 5.10: (a) Bez pogreški; (b) Sa greškom na šestom bitu

Druga situacija sa dvije pogreške znači da smo primili riječ:

$$0\ 1\ 0\ \mathbf{1}\ \mathbf{0}\ 0\ 1\ 0.$$

Pozicija 1: $0+0+0+1=1$ ✗ Pozicija 2: $0+1+0+1=0$ ✓

Pozicija 4: $1+0+0+1=0$ ✓ Pozicija 8: $0+1+0+1+0+0+1+1=0$ ✓.

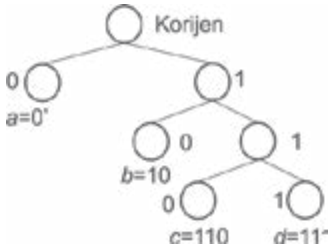
Prva tri sindromska bita ukazuju da se greška dogodila (na prvoj poziciji, što je pogrešno), a posljednji bit kaže da nema greške, odnosno preciznije da nema jedna pogreška. Zaključak je da su detektovane dvije pogreške.

5.12. Alfabet se sastoji od simbola $\{a, b, c, d\}$ koji se pojavljuju sa vjerovatnoćama $\{0.5, 0.3, 0.15, 0.05\}$, respektivno. Kodirati Hafmenovim kodom riječ *bbda*, pa zatim predmetnu riječ kodirati pravougaonim kodom (9,4) sa bitovima parnosti postavljenim na posljednjim pozicijama u kodu. Ukoliko je potrebno, za kodiranje ove riječi izvršiti dopunjavanje nulama. Na 3. poziciji, u prenosu kroz kanal, došlo je do pogreške. Izvršiti dekodiranje do poslate riječi osnovnog alfabeta.

Rješenje: Na Slici V.15 prikazano je stablo Hafmenovog koda za dati alfabet. Data poruka se kodira kao (dodali smo bjeline između kodova pojedinih simbola radi lakšeg vizuelnog razgraničenja mada tako nešto ne postoji u stvarnosti):

$$10\ \ 10\ \ 111\ 0.$$

Imamo 8 informacionih simbola koje dijelimo u dvije poruke 1010 i 1110, koje kodiramo (9,4) pravougaonim kodom. Za prvu sekvencu, biti parnosti su, redom:



Slika V.15. Hafmenov kod za alfabet iz zadatka 5.12

11 (po vrsti), 00 (po koloni) i 0 (za čitavu riječ), dok su za drugu sekvencu biti parnosti, redom: 01 (po vrsti), 01 (po koloni) i 1 za čitavu sekvencu. Stoga dobijamo:

$$101011000\ 111001011.$$

Ako se greška dogodila na trećem bitu, primamo poruku:

$$10\mathbf{0}011000\ 111001011.$$

Kod prve sekvence imamo da biti parnosti po prvoj vrsti i drugoj koloni nisu zadovoljeni (kao i onaj za čitavu riječ), pa se dekodiranje obavlja bez problema, dok kod druge sekvence nema nikakvih problema. Dekoder kanala dekodira:

$$10101110,$$

a zatim se putem Hafmenovog dekodera (stabla) dekodira sa Slike V.15 (dekoder izvora):

$$bbda,$$

a što predstavlja ispravnu poruku.

5.13. Date su kodne riječi 6 kodova. Odrediti minimalno Hemingovo rastojanje i na osnovu toga procijeniti kakve su sposobnosti koda u ispravljanju i detektovanju pogreški.

Rješenje: (a) Ovdje je minimalno rastojanje dvije kodne riječi 1. Znači, kod je jednoznačno dekodabilan. (b) Minimalno Hemingovo rastojanje dvije riječi kod ovog koda je 2. Kod može detektovati jednu pogrešku. (c) Ovdje je rastojanje 3.

Kod (a)	**Kod (b)**	**Kod (c)**	**Kod (d)**	**Kod (e)**	**Kod (f)**
00, 01, 10, 11	000, 011, 101, 110	000, 111	0000, 1111	00000, 11111	00000, 01101, 10110, 11001

Tabela V.6. Kodovi za određivanje minimalnog Hemingovog rastojanja u zadatku 5.13

Kod je u stanju da ispravi jednu pogrešku. (d) Rastojanje je 4. Kod može da ispravi jednu pogrešku i da detektuje dvije. (e) Rastojanje je 5. Kod je u stanju da ispravi dvije pogreške. (f) Minimalno Hemingovo rastojanje je 3, a to znači da je kod u stanju da ispravi jednu pogrešku.

5.14. (a) Koliko je Hemingovo rastojanje između dva vektora koji predstavljaju susjedna tjemena n-dimenzione sfere? (b) Kolika je Hemingova distanca između poruke **x** dužine n i njoj komplementarne poruke?

Rješenje: (a) Hemingovo rastojanje je 1 jer se susjedna tjemena mogu razlikovati na najviše jednoj poziciji. (b) Hemingovo rastojanje je n jer se poruke razlikuju na svim pozicijama.

5.15. Informacioni bit se dijeli u blokove od po 10 bita. Blokove je potrebno kodirati odgovarajućim blok kodovima. Provjerite na koji se najadekvatniji način mogu kodirati ovi blokovi putem pravougaonog, trougaonog i Hemingovog koda, te sračunati i porediti vjerovatnoće greške u postupku dekodiranja.

Rješenje: Kod pravougaonog koda možemo da podijelimo 10 bita u tabeli 2×5, a broj bita parnosti u tom slučaju je 8, jer bi dimenzije koda bile tada $3 \times 6 = 18$ ($18 - 10 = 8$ redundantnih bita). Kod trougaonog koda situacija je čista jer možemo konstruisati (15,10) kod. Kod Hemingovog koda imamo kombinaciju (15,11), ali je možemo svesti na (14,10), ne prenoseći bit koji ne postoji u riječi. Dakle, u prvom slučaju imamo 8, u drugom 5, a u trećem 4 redundantna bita. Pod greškom u navedenim postupcima možemo smatrati situaciju da se dogodi 2 i više pogreški. Pretpostavimo da je $p \ll 1$, pa tada možemo svesti grešku u dekodiranju na slučaj dvije pogreške. Tada je greška kod predmetnih kodova, redom:

$$P_{\text{prav}} \approx \binom{18}{2} p^2 (1-p)^{16} = 153 p^2 (1-p)^{16} \quad P_{\text{troug}} \approx \binom{15}{2} p^2 (1-p)^{13} = 105 p^2 (1-p)^{13}$$

$$P_{\text{ham}} \approx \binom{14}{2} p^2 (1-p)^{12} = 91 p^2 (1-p)^{12}.$$

Sada jasno vidimo da čak i u ovom najpovoljnijem slučaju (veoma male vjerovatnoće greške sa zanemarivanjem mogućnosti pojava više pogreški) imamo da su vjerovatnoće neispravnog rada kod dužih kodova znatno veće nego kod kraće kodne riječi, uz istu sposobnost prenosa informacionih bita (vjerovatnoće greške se odnose u ovom slučaju kao 153:105:91).

V.5.2 Softverska realizacija

A. Napisati program koji formira kontrolnu cifru kod JMBG koda. A zatim napisati naredbe kojima se provjerava da li je neki kod ispravan.

```
clear
x=[0 5 0 2 9 8 6 2 1 0 0 3];
z=[7:-1:2 7:-1:2];
rm=rem(sum(x.*z),11);
if(rm==1 | rm==0)
   c=0;
else
   c=11-rm;
end
y=[x,c];
t=[z,1];
if(rem(sum(y.*t),11)==0 | rem(sum(y.*t),11)==10)
   disp('Ispravan')
else
   disp('Neispravan')
end
y1=y;
y1(6:7)=y([7 6]);
if(rem(sum(y1.*t),11)==0 | rem(sum(y1.*t),11)==10)
   disp('Ispravan')
else
   disp('Neispravan')
end
y1=y;
y1(4:5)=y([5 4]);
if(rem(sum(y1.*t),11)==0 | rem(sum(y1.*t),11)==10)
   disp('Ispravan')
else
   disp('Neispravan')
end
```

Vektor x predstavlja prvih 10 simbola JMBG, dok je vektor z sa težinama. Zatim slijedi proces određivanja kontrolnog bita c, nakon čega provjeravamo da li je riječ ispravna. Zatim dva puta permutujemo cifre broja: jednom šestu i sedmu, a drugi put četvrtu i petu. Ako provjerite, program će u prvom slučaju pogriješiti i proglasiti riječ (kod) ispravnim. Zbog čega? U drugom slučaju kodni postupak ispravno nalazi da je kod nekorektan.

B. Dati su informacioni biti 1001. Izvršiti kodiranje putem pravougaonog (9,4) koda sa bitima parnosti formiranim red po red. Zatim promijenite bit na četvrtoj poziciji i izvršite dekodiranje.

```
inform=[1 0;0 1];
Kod=zeros(3,3);
Kod(1:2,1:2)=inform;
```

```
Kod(1,3)=rem(sum(Kod(1,1:2)),2);
Kod(2,3)=rem(sum(Kod(2,1:2)),2);
Kod(3,1)=rem(sum(Kod(1:2,1)),2);
Kod(3,2)=rem(sum(Kod(1:2,2)),2);
Kod(3,3)=rem(sum(sum(Kod(1:2,1:2))),2);
>> Kod
    1    0    1
    0    1    1
    1    1    0
Rij=Kod';
Rij(:)'
    1    0    1    0    1    1    1    1    0
Rij(4)=1-Rij(4);
Kod(:)=Rij;
Kod =
    1    1    1
    0    1    1
    1    1    0
p2=find(rem(sum(Kod),2)==1)
p1=find(rem(sum(Kod'),2)==1)
Kod(p1,p2)=1-Kod(p1,p2);
Dekod=Kod(1:2,1:2);
Dekod =
    1    0
    0    1
```

Formirali smo matricu dimenzija 3×3, a zatim smo dobili odgovarajuće kontrolne bite. Na primjer, sum(Kod(1,1:2)) je broj jedinica u prvoj vrsti, dok je rem(sum(Kod(1,1:2)),2) bit koji treba dodati u prvi red. Rij je kodna riječ dobijena iz matrice, red po red. Zatim smo izvršili promjenu jednog bita i provjerom bitova parnosti našli poziciju gdje se desila navedena greška. Nakon ispravke greške, dobijena dekodirana poruka Dekod ista je kao polazna.

C. Napisati naredbe kojima se određuje Hemingova težina kodne riječi i Hemingova distanca između dvije kodne riječi.

```
z=rand(1,12)>0.5
    1    1    1    0    1    0    1    0    0    0    0    1
sum(z)
    6
y=rand(1,12)>0.5
y =
    1    0    1    0    0    0    1    1    0    0    0    1
sum(y)
    5
```

```
sum(xor(z,y))
    3
```

Formirali smo dvije binarne riječi dužine 12 bita, z i y. Prostim sumiranjem dobijamo Hemingove težine (u primjeru 6 i 5, respektivno), dok smo sa sum(xor(z,y)) sračunali Hemingovu distancu. Ovdje smo upotrijebili naredbu xor, koja vrši operaciju ekskluzivno ili po elementima vektora argumenata (mada smo mogli da koristimo naredbu rem).

D. Demonstrirati kodiranje i dekodiranje Hemingovim proširenim (8,4) kodom za tri slučaja (nema pogrešaka, jedna pogreška, dvije pogreške).

```
x=[1 1 0 1];
y=zeros(1,8);
y([3 5 6 7])=x;
y(1)=rem(sum(y([3 5 7])),2);
y(2)=rem(sum(y([3 6 7])),2);
y(4)=rem(sum(y([5 6 7])),2);
y(8)=rem(sum(y(1:7)),2);
```

Informacioni biti su smješteni u vektor x. Kodna riječ y formirana je tako što su pravilne binarne pozicije (1, 2, 4) sačuvane za bite parnosti, dok je posljednja pozicija (osma) ostavljena za provjeru parnosti za čitavu riječ. Nakon toga, sumirani su biti na odgovarajućim pozicijama, pa je ostatak, pri dijeljenju sa dva, upotrijebljen za određivanje bita parnosti. Pozovimo sada funkciju sindrom, koja određuje sindrom primljene riječi:

```
S=sindrom(y);
Funkcija sindrom u MATLAB-u ima sljedeći oblik:
function S=sindrom(y)
S(1)=rem(sum(y([1 3 5 7])),2);
S(2)=rem(sum(y([2 3 6 7])),2);
S(3)=rem(sum(y(4:7)),2);
S(4)= rem(sum(y(1:8)),2);
```

U konkretnom slučaju, dobijena vrijednost sindroma je nul-vektor:

```
S=
0 0 0 0
```

što znači da nema pogrešku u primljenoj riječi.

Neka je sada primljena kodna riječi izmijenjena na poziciji tri (odradićemo kao formu negacije):

```
y1=y;
y1(3)=1-y1(3);
```

```
S1=sindrom(y1);
```
Sindrom je sada:
```
S1=
1 1 0 1
```

što ukazuje na to da se greška dogodila na poziciji broj 3 na osnovu prva tri bita sindroma, dok posljednji bit potvrđuje da se greška dogodila. Ako je potrebno da dobijemo eksplicitnu mogućnost ispravke greške, odnosno poziciju pogreške, to se u predmetnom slučaju može uraditi iskazom:

```
sum(S1(1:end-1).*[1 2 4])
```

Konačno, ako dodamo još jednu pogrešku:

```
y2=y1;
y2(6)=1-y2(6);
S2=sindrom(y2);
```
dobijamo sindrom:
```
S2=
1 0 1 0
```

gdje nas prva tri bita navode na pogrešan zaključak o grešci na jednom (petom) bitu, ali nas posljednji bit sindroma opominje da imamo više od jedne pogreške (odnosno, paran broj pogreški).

POGLAVLJE VI

BLOK KODOVI

VI. BLOK KODOVI

Već smo se upoznali sa blok kodovima. Međutim, uveli smo ih intuitivno. U tom smislu nepraktični su za analizu, proširenje i generalizaciju. Neki od stavova koje smo prethodno prikazali su veoma rigidni – na primjer, postavljanje kontrolnih bita u kodnoj riječi za Hemingove kodove na pravilnim binarnim pozicijama. Srećom, uveli smo mnogo teorijski bitnih elemenata, kao što je Hemingova distanca, tako da smo sada u stanju da, korišćenjem dva alata, ponovo posjetimo blok kodove. Prvo ćemo uvesti kodiranje i dekodiranje putem matrica. Biće uvedene **kontrolna** i **generatorska matrica**, te **sindrom** kao indikator „bolesti" naše kodne riječi. Kratko će u Sekciji VI.2 biti objašnjen problem koji se dešava u uslovima pojave većeg broja pogreški u jednoj kodnoj riječi, te će biti uvedeni blokovi **interliver** i **deinterliver** koji (djelimično) mogu da ublaže ovaj problem. U Sekciji VI.3, nakon što elaboriramo mane matričnog pristupa, uvodi se najmoćniji alat **algebarske kodne teorije**, odnosno prikaz kodova putem **polinoma**. Između ostalih vrlina, polinomijalni pristup omogućava dizajniranje efikasnih hardverskih šema, pa će biti uvedena dva tipa kodera ovih kodova zasnovanih na **pomjeračkom registru** (u jednom slučaju sa **povratnom spregom**). Nakon toga prelazimo na proučavanje nebinarnih kodova za ispravljanje pogreški u sistemu. Posebno interesantan problem, vezan za ispravljanje više grešaka na principima Hemingovih kodova, opisan je u Sekciji VI.5. Ukazano je da generalizacija ovih kodova na problem ispravke više od jedne pogreške nije trivijalna, pa za tu namjenu u Sekciji VI.6 dajemo opis **BCH kodova** (engl. *Bose–Chaudhuri–Hocquenghem*).

VI.1 Blok kodovi u matričnom obliku

Blok kodove smo označavali sa (n,k), gdje je n broj bita u kodnoj riječi, od kojih je k informacionih. Kodni odnos ovih kodova je $R = n/k$. Za formulisanje bitnih karakteristika ovih kodova možemo posmatrati bilo koji od uvedenih kodova.

Za primjer uzmimo Hemingov (7,4) kod sa bitovima parnosti raspoređenim na pravilnim binarnim pozicijama:

$$[c_1\ c_2\ i_1\ c_3\ i_2\ i_3\ i_4].$$

U datom vektoru *c*-ovi predstavljaju bitove provjere parnosti, dok su *i* informacioni biti. Bit parnosti na prvoj poziciji kontroliše bitove na pozicijama {1,3,5,7}, bit parnosti sa pozicije 2 kontroliše bite na pozicijama {2,3,6,7}, dok bit parnosti sa pozicije 4 kontroliše pozicije {4,5,6,7}. Ako je primljena ispravna kodna riječ (bez greške u kanalu), onda će biti zadovoljene sljedeće tri jednačine:

$$\begin{aligned} c_1 + \quad i_1 + \ &+ i_2 + \quad i_4 = 0 \\ c_2 + i_1 \quad &+ i_3 + i_4 = 0 \\ c_3 + i_2 + i_3 + i_4 &= 0. \end{aligned}$$

Naravno, operacija + je, zapravo, ekskluzivno ili, odnosno sabiranje po modulu 2. Ova relacija se može napisati na još upečatljiviji način:

$$\begin{aligned} 1c_1 + 0c_2 + 1i_1 + 0c_3 + 1i_2 + 0i_3 + 1i_4 &= 0 \\ 0c_1 + 1c_2 + 1i_1 + 0c_3 + 0i_2 + 1i_3 + 1i_4 &= 0 \\ 0c_1 + 0c_2 + 0i_1 + 1c_3 + 1i_2 + 1i_3 + 1i_4 &= 0, \end{aligned}$$

odnosno u matričnom obliku:

$$\mathbf{c}\mathbf{H}^{\mathrm{T}} = \mathbf{0},$$

gdje je **c** kodna riječ $\mathbf{c} = [c_1\ c_2\ i_1\ c_3\ i_2\ i_3\ i_4]$, **0** je vektor kolona sa tri nule, dok se **H** naziva kontrolnom matricom Hemingovog koda, koja je u predmetnom slučaju:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Odmah se može uočiti da su dimenzije kontrolne matrice jednake $(n-k)\times n = m\times n$, odnosno broj vrsta je jednak broju redundantnih kontrolnih bita, dok je broj kolona jednak dužini kodne riječi. Prikažimo kontrolnu matricu za nekoliko karakterističnih kodova.

a. Kod sa provjerom parnosti (8,7):

$$\mathbf{H} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1].$$

b. Repetitivni kod (3,1) (sami protumačite ovo u smislu Hemingovog koda koji je u stanju da ispravi jednu pogrešku):

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

c. Pravougaoni kod (9,4). Kodna riječ je formirana vrstu po vrstu:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Posljednji bit u ovoj kombinaciji je, kao što smo već ukazali, najintrigantniji. On provjerava čitavu kodnu riječ (mogle su stajati sve jedinice u posljednjoj vrsti), ali je u slučaju ovog koda to ekvivalentno provjeri samo informacionih bita.

d. Trougaoni (10,6) kod. Kodna riječ je i ovdje formirana vrstu po vrstu:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

e. Prošireni Hemingov kod (8,4) sa bitima parnosti pa pozicijama 1, 2 i 4 i bitom parnosti koji kontroliše čitavu kodnu riječ na poziciji 8:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Vratimo se na kontrolnu matricu Hemingovog koda (7,4). Uočimo da se kod nje pojavljuju sve moguće trobitne nenulte kolone ($2^3 - 1 = 7$). Slično bi se kod (15,11) koda pojavile sve moguće nenulte kombinacije od 4 bita, što rezultuje kontrolnom matricom dimenzija $(2^4 - 1) \times 4 = 15 \times 4$.

Posmatrajmo sada šta se dešava u slučaju kada se primi kodna riječ u kojoj postoji greška. Označimo sada ovu primljenu kodnu riječ kao **r**, a nju možemo prikazati kao ispravnu kodnu riječ **c** kojoj se superponira riječ koja predstavlja grešku **e**:

$$\mathbf{r} = \mathbf{c} + \mathbf{e},$$

gdje je sa $+$ označeno ekskluzivno ili po bitovima. Na poziciji gdje nema greške elementi vektora **e** su nula, tako da se simbol kodne riječi ne mijenja, dok su na pozicijama greške simboli vektora **e** jednaki 1, prouzrokujući promjenu kodne riječi. Pretpostavimo da postoji samo jedna greška, odnosno samo jedna jedinica u vektoru **e**:

$$w_H(\mathbf{e}) = 1.$$

Odredimo čemu je jednako:

$$\mathbf{S} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = (\mathbf{c}+\mathbf{e})\mathbf{H}^{\mathrm{T}} = \mathbf{c}\mathbf{H}^{\mathrm{T}} + \mathbf{e}\mathbf{H}^{\mathrm{T}} = \mathbf{0} + \mathbf{e}\mathbf{H}^{\mathrm{T}} = \mathbf{e}\mathbf{H}^{\mathrm{T}}.$$

Vidimo da ovaj izraz, označen sa **S**, zavisi samo od vektora greške. U slučaju da greške nema, **S** je jednako **0**, dok u slučaju postojanja greške dobijamo da je $\mathbf{S} \neq \mathbf{0}$ i zavisi samo od vektora greške, a ne i od poruke koja je poslata. Stoga se vektor **S** naziva **sindrom** jer ukazuje na poziciju gdje je naš kod „bolestan“, odnosno na poziciju gdje se greška dogodila. Kako se u predmetnom primjeru dogodila greška na jednom bitu, ovaj vektor je sa samo jednim elementom $e_j = 1$ (j je pozicija greške), dok su svi ostali elementi vektora jednaki nuli. Množenjem ovog vektora sa $\mathbf{H}^{\mathrm{T}}$ dolazi do množenja samo jedinice sa j-te pozicije s elementima j-te vrste matrice $\mathbf{H}^{\mathrm{T}}$ (odnosno j-te kolone matrice **H**). Stoga je, u slučaju pojave jedne greške, sindrom jednak:

$$\mathbf{S} = \mathbf{h}_j,$$

gdje je $\mathbf{h}_j$ j-ta kolona matrice **H**. Dakle, sindrom je jednak koloni matrice na poziciji na kojoj se pogreška dogodila. Stoga se na osnovu sindroma može jednoznačno odrediti pozicija jedne greške, u slučaju da se jedna pogreška desila. Algoritam je, kao što vidimo, veoma jednostavan: u slučaju nultog sindroma podrazumijevamo da nema greške u sistemu, dok se u slučaju nenultog sindroma greška dogodila na onoj lokaciji koja je jednaka koloni kontrolne matrice. Kod binarnih kodova situacija je prosta kada znamo gdje se greška dogodila jednostavnom negacijom primljenog bita vršimo korekciju pogreške.

Primjer VI.1. Uzmimo za primjer (9,4) pravougaoni kod, gdje su informacioni biti:

$$\begin{matrix} 1 & 0 \\ 1 & 1. \end{matrix}$$

Primjenjujući postupak formiranja bitova parnosti, dobijamo sljedeće bite kodne riječi:

$$\begin{matrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1. \end{matrix}$$

Kodnu riječ formiramo red po red, pa dobijamo:

$$1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1.$$

Pretpostavimo da se greška dogodila na 4. bitu, pa je primljena poruka:

$$1 \quad 0 \quad 1 \quad \underline{\mathbf{0}} \quad 1 \quad 0 \quad 0 \quad 1 \quad 1.$$

Sračunajmo sada sindrom:

$$\mathbf{S} = \mathbf{r}\mathbf{H}^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \mathbf{0} & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Očigledno, greška se dogodila na 4. lokaciji jer je dobijeni sindrom jednak (transponovanoj) 4. koloni kontrolne matrice. Sada znamo da je na toj poziciji umjesto primljenog bita 0, ispravan bit 1. □

U slučaju koda koji je u stanju da ispravi jednu i detektuje dvije pogreške, kao što je Hemingov (8,4) kod, ništa se krupno ne mijenja u slučajevima kada nema grešaka (sindrom jednak nula vektoru) i kada postoji jedna greška (sindrom je jednak koloni koja postoji u kontrolnoj matrici na poziciji koja odgovara grešci u primljenoj riječi). Šta se dešava kada imamo dvije pogreške? Sindrom je u ovom slučaju jednak sumi dviju kolona u kojima su se greške desile:

$$\mathbf{S} = \mathbf{h}_i + \mathbf{h}_j,$$

gdje su ovdje i-ta i j-ta pozicija mjesta gdje su se pogreške dogodile. Neka je u našem slučaju $i = 4$ i $j = 7$. Dobijamo da je sindrom sada:

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}.$$

Dobili smo nenulti sindrom koji nije jednak nijednoj koloni kontrolne matrice. Na osnovu ovoga zaključujemo da se dogodilo više pogreški (u konkretnom slučaju dvije).

Već smo kod nekih kodova uočili da smo slobodno sami određivali na kojem mjestu u kodnoj riječi da se postave bitovi parnosti. To je isto moguće uraditi i kod

Hemingovog koda. Do sada smo posmatrali bite parnosti raspoređene na pravilnim binarnim pozicijama, s kontrolnom matricom:

$$\mathbf{H}=\begin{bmatrix}1&0&1&0&1&0&1\\0&1&1&0&0&1&1\\0&0&0&1&1&1&1\end{bmatrix}.$$

Ova kontrolna matrica odgovara kodnoj riječi:

$$\mathbf{c}=[c_1 \quad c_2 \quad i_1 \quad c_3 \quad i_2 \quad i_3 \quad i_4].$$

Ako preuredimo (permutujemo) ovu kontrolnu riječ tako da kodni bitovi provjere parnosti c_1, c_2 i c_3 dođu na njen kraj, nakon informacionih bita i_1, i_2, i_3 i i_4:

$$\mathbf{c}=[i_1 \quad i_2 \quad i_3 \quad i_4 \quad c_1 \quad c_2 \quad c_3],$$

onda moramo i kontrolnu matricu da preuredimo tako da odgovara ovoj kodnoj riječi, permutujući kolone na isti način kako su permutovani biti u kodnoj riječi:

$$\mathbf{H}=\begin{bmatrix}1&1&0&1&1&0&0\\1&0&1&1&0&1&0\\0&1&1&1&0&0&1\end{bmatrix}.$$

Slobodni smo da permutujemo svaki kod na proizvoljan način, ali s tim da kontrolnu matricu modifikujemo na isti način kako je to urađeno i kod kodne riječi.

Možemo uočiti i jednu veoma bitnu osobinu Hemingovog koda i njegove kontrolne matrice. Vidimo da ova kontrolna matrica (u datom slučaju) posjeduje sve nenulte kolone koje se mogu konstruisati na osnovu tri bita (tri kontrole parnosti). Na isti način, za Hemingov (15,11) kod kontrolna matrica sadrži sve nenulte kolone koje se mogu konstruisati sa četiri bita (četiri kontrole parnosti). U generalnom slučaju, za Hemingov kod (n,k) kontrolna matrica sadrži sve nenulte (različite) kolone koje se mogu konstruisati sa $n-k$ elemenata, odnosno $2^{n-k}-1$ kolone. Dakle, Hemingov kod se može uvesti na nešto sistematičniji način nego što smo to uradili prethodno. Prvo smo uočili da više ne postoji potreba da se kontrolni biti nalaze na pravilnim binarnim pozicijama, već da ih možemo rasporediti po želji. Druga činjenica, koja se takođe može jednostavno uočiti, jeste da je broj kolona kontrolne matrice jednak $2^{n-k}-1$, što znači da je:

$$2^{n-k}-1=n.$$

Odavde slijedi:

$$n-k=\log_2(n+1)$$
$$k=n-\log_2(n+1).$$

Kontrolna matrica Hemingovog koda (15,11) sa bitovima parnosti na posljednjim mjestima u riječi, može da ima oblik:

$$\mathbf{H}=\begin{bmatrix}1&1&1&1&1&1&1&0&0&0&0&1&0&0&0\\1&1&1&1&0&0&0&1&1&1&0&0&1&0&0\\1&1&0&0&1&1&0&1&1&0&1&0&0&1&0\\1&0&1&0&1&0&1&1&0&1&1&\mathbf{0}&0&0&1\end{bmatrix}.$$

Uočimo da se ovdje ponavlja situacija da na kraju kontrolne matrice imamo jediničnu matricu dimenzija $(n-k)\times(n-k)$. Ovakvi kodovi se nazivaju **sistematski** i njihove kontrolne matrice se mogu prikazati u obliku:

$$\mathbf{H}=\left[\mathbf{P}\,|\,\mathbf{I}_{n-k}\right],$$

gdje je $\mathbf{P}$ matrica dimenzija $(n-k)\times k$, dok je $\mathbf{I}_{n-k}$ jedinična matrica dimenzija $(n-k)\times(n-k)$. Ovakav, često pogodan način organizacije kodova nije ograničen samo na Hemingov kod već se može primijeniti na bilo koji od uvedenih kodova, te na druge blok kodove.

Primjer VI.2. Prikazati kontrolnu matricu pravougaonog (9,4) koda za slučaj kada je kodna riječ formirana tako da su kontrolni biti postavljeni na posljednjim pozicijama u kodnoj riječi. Formirati kodnu riječ za informacione bite 0101. Unijeti grešku na petom bitu, odrediti sindrom i korigovati poruku.

Rješenje: Kodna riječ je, u ovom slučaju, oblika:

$$\mathbf{c}=[i_1\ i_2\ i_3\ i_4\ c_1\ c_2\ c_3\ c_4\ c_5].$$

Kontrolna matrica, u ovom slučaju, ima oblik:

$$\mathbf{H}=\begin{bmatrix}1&1&0&0&1&0&0&0&0\\0&0&1&1&0&1&0&0&0\\1&0&1&0&0&0&1&0&0\\0&1&0&1&0&0&0&1&0\\1&1&1&1&0&0&0&0&1\end{bmatrix}.$$

Za informacione bite 0101, kontrolne bite (označene boldirano) možemo dobiti na sljedeći način:

$$\begin{matrix}0&1&\mathbf{1}\\0&1&\mathbf{1}\\\mathbf{0}&\mathbf{0}&\mathbf{0},\end{matrix}$$

pa je kodna riječ $\mathbf{c} = [0\ 1\ 0\ 1\ \mathbf{1\ 1\ 0\ 0\ 0}]$. Primljena riječ sa greškom na petom bitu je: $\mathbf{r} = [0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0]$. Sračunajmo sada sindrom:

$$\mathbf{S} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = [0\,1\,0\,1\,0\,1\,0\,0\,0]\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & \mathbf{0} & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Dakle, dobili smo nenulti sindrom koji je jednak petoj koloni, na osnovu čega se sada može ispraviti odgovarajuća greška.

Kontrolna matrica koda se može upotrijebiti da bi se putem nje odredilo minimalno Hemingovo rastojanje u kodu. Uočimo da je kod sa svim nulama zasigurno ispravna riječ ovog koda, jer ako postavimo nule u informacionim bitima, jedini način da zadovoljimo relacije parnosti je da su nule postavljene i na kontrolnim pozicijama. Zapis koji imamo kod kontrolne matrice u suštini uvijek možemo da riješimo po kontrolnim bitima. Na primjer, za Hemingov kod (7,4), sa bitovima parnosti na pravilnim binarnim pozicijama, važi:

$$\begin{aligned} &c_1 + i_1 + i_2 + i_4 = 0 \quad c_1 + c_1 + i_1 + i_2 + i_4 = c_1 \quad i_1 + i_2 + i_4 = c_1 \\ &c_2 + i_1 + i_3 + i_4 = 0 \Rightarrow c_2 + c_2 + i_1 + i_3 + i_4 = c_2 \Rightarrow i_1 + i_3 + i_4 = c_2 \\ &c_3 + i_2 + i_3 + i_4 = 0 \quad c_3 + c_3 + i_2 + i_3 + i_4 = c_3 \quad i_2 + i_3 + i_4 = c_3. \end{aligned}$$

Postavlja se pitanje: kako korišćenjem matrične algebre možemo da prikažemo generisanje kodne riječi? Ako sa **i** označimo vektor informacionih bita, tada se generisana kodna riječ može sračunati kao:

$$\mathbf{c} = \mathbf{i}\mathbf{G},$$

gdje se **G** naziva generatorskom matricom koda. Ako je kodna riječ blok koda (n,k) dimenzije $1\times n$, a vektor informacionih bita dimenzije $1\times k$ (broj informacionih bita je k), da bi bili zadovoljeni uslovi matričnog množenja, dimenzija generatorske matrice **G** mora biti $k\times n$. Kao primjer posmatrajmo slučaj Hemingovog koda sa bitima parnosti raspoređenim na pravilnim binarnim pozicijama, gdje je kodna riječ:

$$\mathbf{c} = [c_1 \quad c_2 \quad i_1 \quad c_3 \quad i_2 \quad i_3 \quad i_4].$$

Problem koji želimo da riješimo jeste da odredimo koeficijente generatorske matrice koja na osnovu informacionog vektora $\mathbf{i} = [i_1\ \ i_2\ \ i_3\ i_4]$ produkuje odgovarajuću kodnu riječ. Odnosno, ovo možemo zapisati kao:

$$[c_1 \quad c_2 \quad i_1 \quad c_3 \quad i_2 \quad i_3 \quad i_4] = [i_1 \quad i_2 \quad i_3 \quad i_4] \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} & g_{15} & g_{16} & g_{17} \\ g_{21} & g_{22} & g_{23} & g_{24} & g_{25} & g_{26} & g_{27} \\ g_{31} & g_{32} & g_{33} & g_{34} & g_{35} & g_{36} & g_{37} \\ g_{41} & g_{42} & g_{43} & g_{44} & g_{45} & g_{46} & g_{47} \end{bmatrix}.$$

Kao što će se vidjeti, premda izgleda komplikovano, situacija je mnogo prostija nego ovako zapisana relacija. Naime, kontrolni bit c_1 je formiran da zadovolji provjeru parnosti na neparnim pozicijama u kodnoj riječi, odnosno zadovoljava:

$$c_1 + i_1 + i_2 + i_4 = 0.$$

Dodajmo sada i na lijevu i na desnu stranu ove relacije c_1:

$$c_1 + c_1 + i_1 + i_2 + i_4 = c_1 + 0.$$

Kod operacije ekskluzivno ili važi $0+0=0$, $1+1=0$, odnosno $c_1 + c_1 = 0$, takođe $c_1 + 0 = c_1$, pa dobijamo:

$$c_1 = i_1 + i_2 + i_4 = i_1 + i_2 + 0i_3 + i_4.$$

Dakle, da bismo dobili da je na prvom mjestu kodne riječi kontrolni simbol c_1, po prethodnoj formuli slijedi da vektor informacionih bita $\mathbf{i} = [i_1 \ i_2 \ i_3 \ i_4]$ mora biti pomnožen sa vektor kolonom $[1\ 1\ 0\ 1]^{\mathrm{T}}$ kako bi produkovala c_1, iz čega slijedi da je prva kolona matrice **G**: $[g_{11} \ g_{21} \ g_{31} \ g_{41}]^{\mathrm{T}} = [1\ 1\ 0\ 1]^{\mathrm{T}}$. Na isti način slijedi da je naredna kolona dobijena iz relacije:

$$c_2 + i_1 + i_3 + i_4 = 0 \Rightarrow c_2 = i_1 + i_3 + i_4 \Rightarrow [g_{12} \ g_{22} \ g_{32} \ g_{42}]^{\mathrm{T}} = [1\ 0\ 1\ 1]^{\mathrm{T}}.$$

Kod informacionih bita stvar je još jednostavnija jer se, na primjer, može zapisati za prvi informacioni bit koji se nalazi na trećoj poziciji da je:

$$i_1 = [i_1 \quad i_2 \quad i_3 \quad i_4][g_{13} \quad g_{23} \quad g_{33} \quad g_{43}]^T = [i_1 \quad i_2 \quad i_3 \quad i_4][1 \quad 0 \quad 0 \quad 0]^T \Rightarrow$$
$$[g_{13} \quad g_{23} \quad g_{33} \quad g_{43}] = [1 \quad 0 \quad 0 \quad 0].$$

Dakle, na osnovu navedenog nije teško konstruisati generatorsku matricu ovoga (kao i ostalih blok kodova):

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Uzmimo sada pravougaoni kod (9,4) (razmatran u Primjeru VI.2) sa kodnom riječju:

$$\mathbf{c} = [i_1\ i_2\ i_3\ i_4\ c_1\ c_2\ c_3\ c_4\ c_5],$$

gdje su biti parnosti postavljeni na posljednja mjesta u kodnoj riječi. Generatorska matrica ovog koda je dimenzija 4×9 i može se formirati kao:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Uočimo da smo ovdje dobili da je na početku matrice formirana jedinična matrica, dok je ostatak formiran u drugom obliku. Predmetnu generatorsku matricu ovog koda sada možemo zapisati kao:

$$\mathbf{G} = \left[\mathbf{I}_k \mid \mathbf{R}\right].$$

Sada je dobar momenat da se istraži matrica **R**, odnosno da se utvrdi kakva je veza ove matrice sa matricom **P** kod kontrolne matrice. Uvrštavajući kodnu riječ $\mathbf{c} = \mathbf{iG}$ u izraz za sindrom, dobijamo:

$$\mathbf{S} = \mathbf{cH}^{\mathrm{T}} = (\mathbf{iG})\mathbf{H}^{\mathrm{T}} = \mathbf{i}(\mathbf{GH}^{\mathrm{T}}).$$

Dakle, da bismo dobili da je za svaku informacionu riječ sindrom, kada nema greške u kodnoj poruci, jednak nula vektoru $\mathbf{S} = \mathbf{0}$, neophodno je da važi:

$$\mathbf{GH}^{\mathrm{T}} = \mathbf{0}.$$

U slučaju kada su informacioni simboli postavljeni na početak kodne riječi, važi:

$$\mathbf{GH}^{\mathrm{T}} = \left[\mathbf{I}_k \mid \mathbf{R}\right]\left[\mathbf{P} \mid \mathbf{I}_{n-k}\right]^{\mathrm{T}} = \left[\mathbf{I}_k \mid \mathbf{R}\right]\begin{bmatrix} \mathbf{P}^{\mathrm{T}} \\ \hline \mathbf{I}_{n-k} \end{bmatrix} = \mathbf{I}_k\mathbf{P}^{\mathrm{T}} + \mathbf{RI}_{n-1} = \mathbf{P}^{\mathrm{T}} + \mathbf{R} = \mathbf{0}.$$

U predmetnoj aritmetici gdje je + operacija ekskluzivno ili, odnosno gdje je sabiranje i oduzimanje ista operacija, slijedi da je:

$$\mathbf{R} = \mathbf{P}^{\mathrm{T}}.$$

Razmislite o vezi koja se može uspostaviti između matrica **G** i **H** za slučaj opšteg rasporeda informacionih i kontrolih bita u kodnoj riječi. Uočimo da je kod ovih kodova kodna riječ sa svim nulama uvijek ispravna, jer kada se nula vektor pomnoži sa generatorskom matricom, sigurno dobijamo nula vektor:

$$\mathbf{0} = \mathbf{0G}.$$

Naravno, rezultujući nula vektor je dužine *n*, dok je nula vektor koji predstavlja informacione bite dužine *k*. Ako imamo dva vektora koji predstavljaju informacione bite $\mathbf{i}_1$ i $\mathbf{i}_2$, tada su odgovarajuće kodne riječi:

$$\mathbf{c}_1 = \mathbf{i}_1\mathbf{G} \qquad\qquad \mathbf{c}_2 = \mathbf{i}_2\mathbf{G}.$$

Kodna riječ koja odgovara zbiru informacionih vektora jednaka je zbiru kodnih riječi $\mathbf{c}_1$ i $\mathbf{c}_2$:

$$(\mathbf{i}_1 + \mathbf{i}_2)\mathbf{G} = \mathbf{i}_1\mathbf{G} + \mathbf{i}_2\mathbf{G} = \mathbf{c}_1 + \mathbf{c}_2.$$

Minimalna Hemingova distanca koda definisana je kao minimalna distanca između bilo koje dvije različite ispravne kodne riječi koda:

$$\min_{\substack{i,j \\ i \neq j}}(d_H(\mathbf{c}_i, \mathbf{c}_j)).$$

Distanca između dvije kodne riječi $\mathbf{c}_i$ i $\mathbf{c}_j$ neće se izmijeniti ako obje kodne riječi saberemo sa nekim konstantnim vektorom:

$$d_H(\mathbf{c}_i, \mathbf{c}_j) = d_H(\mathbf{c}_i + \mathbf{c}, \mathbf{c}_j + \mathbf{c}).$$

Ovo se u principu lako pokazuje. Naime, Hemingova distanca je jednaka sumi Hemingovih distanci po bitovima. Ako je bit u riječi **c** jednak nuli, onda se odgovarajući bitovi u riječima $\mathbf{c}_i$ i $\mathbf{c}_j$ ne mijenjaju, dok ako je u pitanju jedinica, biti u $\mathbf{c}_i$ i $\mathbf{c}_j$ se invertuju, pa ponovo Hemingova distanca ostaje ista kao u originalnoj poruci. Stoga se uvijek može pokazati da je:

$$d_H(\mathbf{c}_i, \mathbf{c}_j) = d_H(\mathbf{c}_i + \mathbf{c}_i, \mathbf{c}_j + \mathbf{c}_i) = d_H(\mathbf{0}, \mathbf{c}_j + \mathbf{c}_i) = w_H(\mathbf{c}_j + \mathbf{c}_i).$$

Kako se sa $\mathbf{c}_i + \mathbf{c}_j$ mogu generisati sve kodne riječi, onda se pokazuje da se minimalna Hemingova distanca može sračunati kao minimum Hemingove težine za sve nenulte kodne riječi, za blok kodove koji su formirani na predmetni način:

$$\min_{\mathbf{c}, \mathbf{c} \uparrow \mathbf{0}}(w_H(\mathbf{c})).$$

Značaj ovog postupka je u tome što je pretraživanje po skupu Hemingovih distanci dvodimenziono, dok je po skupu Hemingovih težina jednodimenziono. Imajući na umu da je broj nenultih ispravnih kodnih riječi $2^k - 1$, te da *k* može biti relativno veliki broj, ovo je značajna ušteda. Može se pokazati da je minimalna distanca blok koda jednaka miminalnom broju kolona kontrolne matrice koje su linearno zavisne. Kolone kontrolne matrice možemo posmatrati kao skup vektora $\mathbf{h}_i$, $i = 1$, 2, ..., *n*. Vektori su linearno zavisni ako postoji težinska suma sa nenultim koeficijentima koja daje nula vektor:

$$\sum_{j=1}^{k} \omega_j \mathbf{h}_{j(i)} = \mathbf{0},$$

gdje makar jedan od koeficijenata ω_i, $i=1, ..., k$ mora biti nenulti. Indeksi $j(i)$ označavaju neku permutaciju kolona matrice **H**. Tako, na primjer, kod Hemingovog (7,4) koda:

$$\mathbf{H}=\begin{bmatrix}1&1&0&1&1&0&0\\1&0&1&1&0&1&0\\0&1&1&1&0&0&1\end{bmatrix}$$

ne postoji nijedna pojedinačna kolona koja je jednaka nuli, niti zbir dvije kolone koji je jednak nuli (jer ne postoje dvije identične kolone). Međutim, sabiranjem prve kolone sa petom i šestom, dobijamo nula kolonu:

$$\begin{bmatrix}1\\1\\0\end{bmatrix}+\begin{bmatrix}1\\0\\0\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}.$$

Dakle, možemo zaključiti da je minimalna distanca ovog koda jednaka 3.

VI.2 Interliver i deinterliver

U praksi, uz rijetke izuzetke, kanali nisu takvi da produkuju greške koje se mogu opisati putem identičnih i na isti način distribuiranih procesa. Nasuprot tome, najčešće u jednom dijelu intervala greške su rijetke ili ih nema, a zatim pod uticajem različitih faktora (npr. vremenskih nepogoda ili specifičnih ljudskih aktivnosti) dolazi do pojave većeg broja grešaka u kratkom vremenskom intervalu. To dovodi do pojave većeg broja pogreški u kodnim riječima. Ako ostanemo na kodu koji je u stanju da ispravi jednu pogrešku po kodnoj riječi, imaćemo određeni broj simbola koji su pogrešno dekodirani. Kako uslovi koji dovode do većeg broja pogreški mogu da potraju, onda sama komunikacija može da bude veoma otežana ili neupotrebljiva. Jedan način da se ovo prevaziđe je da se umjesto kodova koji ispravljaju jednu pogrešku koriste kodovi koji su u stanju da isprave više od jedne pogreške. Međutim, kao što će kasnije biti pokazano, kod ovih kodova moramo da dodamo još provjera parnosti na račun informacionih bita, čime smanjujemo kodni odnos, i samim tim smanjujemo efikasnost korišćenja komunikacionih sredstava, unošenjem veće redundancije u samu poruku, a ovo opet dovodi do finansijske štete po učesnike u komunikaciji. Ovakve kodove uvodimo kasnije u ovom poglavlju.

Interliver je posebni hardverski ili softverski blok koji preuređuje/permutuje simbole uzastopnih kodnih riječi tako da se u nekoliko uzastopnih bita ne nalaze biti iz iste kodne riječi. Na taj način ako se dese greške u uzastopnim bitima poslate riječi, u svakoj pojedinačnoj kodnoj riječi ne nalazi se više od jednog pogršenog bita. Blok na ulazu u prijemnik (prije dekodera kanala) vraća kodne bite u originalni raspored. Predmetni blok na ulazu u dekoder kanala naziva se **deinterliver**.

Posmatrajmo sada sljedeći primjer. Koder kanala kodira Hemingov kod (7,4) sa bitima parnosti raspoređenim na pravilnim binarnim pozicijama. Pretpostavimo da šaljemo ukupno 16 bita:

1 1 0 1 1 0 1 0

0 1 0 1 0 0 1 1.

Dobijene kodne riječi su:

1 1 0 1 ⇒ 1 0 1 0 1 0 1
1 0 1 0 ⇒ 1 0 1 1 0 1 0
0 1 0 1 ⇒ 0 1 0 0 1 0 1
0 0 1 1 ⇒ 1 0 0 0 0 1 1.

Do sada smo smatrali da šaljemo kodne riječi redom (vrstu po vrstu prethodne „matrice“):

1 0 1 0 1 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 1 1.

Međutim, ako se u kanalu dogode uzastopne pogreške na pozicijama, na primjer, od 10 do 13, dobijamo sljedeću poruku:

1 0 1 0 1 0 1 1 0 **0** **0** **1** **0** 0 0 1 0 0 1 0 1 1 0 0 0 0 1 1.

Ispravno bismo dekodirali prvu, treću i četvrtu kodnu riječ, dok bi druga kodna riječ, u kojoj su se dogodile višestruke pogreške, bila pogrešno dekodirana (greške smo označili boldirano i podvučeno da bismo ih lakše razlikovali od pravilno prenesenih bita).

Interliver se uvodi da permutuje kodiranu poruku tako da se biti koji pripadaju istoj kodnoj riječi ne nalaze blizu u poruci koja ide u kanal. Na primjer, u ovom slučaju možemo da pošaljemo prve bite nekoliko kodiranih riječi (u našem slučaju četiri), zatim druge itd., odnosno da šaljemo prethodnu matricu kolonu po kolonu:

1 1 0 1| 0 0 1 0| 1 1 0 0| 0 1 0 0| 1 0 1 0| 0 1 0 1| 1 0 1 1.

Vertikalnim linijama smo odvojili pojedine kolone matrice radi lakšeg praćenja. Pretpostavimo da su se ponovo desile pogreške na 4 uzastopna bita, od 10. do 13:

1 1 0 1| 0 0 1 0| 1 1 **1** **1**| **1** **0** 0 0| 1 0 1 0| 0 1 0 1| 1 0 1 1.

Deinterliving konvertuje poruku u prirodan raspored bita:

1 0 1 **1** 1 0 1
1 0 1 **0** 0 1 0
0 1 **1** 0 1 0 1
1 0 **1** 0 0 1 1.

Vidimo da je u ovom slučaju svaka primljena poruka sa po jednom pogreškom, tako da Hemingov kod (7,4) može uspješno da dekodira ovakvu poruku!

Napominjemo da dizajniranje interlivera i deinterlivera zahtijeva poznavanje kanala, odnosno toga kakve se greške u njemu mogu pojaviti. Interliveri se danas koriste i kod veoma sofisticiranih algoritama za kodiranje kanala, kao što su turbo-kodovi, o kojima će biti više riječi u narednom poglavlju, ali iz drugih razloga, koji će tamo biti i objašnjeni.

VI.3 Definicija kodova putem polinoma

Do sada smo praktično iste kodove uveli dva puta. Prvi put intuitivno, na osnovu elementarne algebre i definicije bitova parnosti. U ovom poglavlju smo pristupili znatno sistematičnije. Uveli smo kodove putem linearne algebre sa kontrolnom, generatorskom matricom i sindromom. Matrični račun je prihvatljiv za jednog inženjera, veoma je razumljiv i lak za programiranje. Međutim, postoje i značajni problemi koji sprečavaju upotrebu matričnog računa u praktičnim aplikacijama kodova kanala. Prvo, ne postoji jednostavan algoritam za generalizaciju ovog postupka za korekciju većeg broja pogreški. Drugi problem je u dimenzijama matrica. Dužina kodne riječi često je reda veličine stotina, pa i hiljada bitova. Kontrolna matrica onda ima velike dimenzije, od nekoliko hiljada kolona sa desetinama vrsta. Još gore, dimenzije generatorske matrice su još veće i reda veličine hiljada redova (vrsta) i hiljada kolona. Ovakva dimenzionalnost povlači značajne memorijske resurse, što može biti kritično kod brojnih sistema za prenos informacija. Na primjer, uređaji na satelitskim sistemima koji obrađuju ogromne količine podataka mogu biti neprihvatljivo skupi ili se ne mogu implementirati na malim raspoloživim resursima (uz činjenicu da bi razvoj čipova ovakvih performansi, koji uz to treba da podnesu veoma teške uslove u svemirskoj eksploataciji, bio veoma skup). Slično je i u prenosu podataka u terestičkim (zemaljskim) uslovima, gdje ovakvi memorijski zahtjevi mogu dovesti do usporavanja sistema, neprihvatljive cijene uređaja itd. Ne treba izgubiti iz vida ni eventualno povećanje složenosti algoritama za implementaciju kodiranja i dekodiranja kada se radi sa ovako velikim generatorskim i kontrolnim matricama. Stoga je bilo od ogromnog značaja da se uvede alternativni koncept, možda manje očigledan, koji bi omogućio generisanje i korekciju kodova sa više grešaka, a sa redukovanim memorijskim i računskim zahtjevima u odnosu na matrični račun. Predmetni alat postoji i sublimiran je **algebarskom kodnom teorijom**. Osnovno sredstvo u okviru algebarske kodne teorije je prikazivanje kodova putem odgovarajućih polinoma.

Kodne riječi za blok kod (n,k) biće generisane na sljedeći način:

$$\mathbf{c}(x) = \mathbf{g}(x)\mathbf{i}(x),$$

gdje je $\mathbf{i}(x)$ informacioni polinom reda $k-1$:

$$\mathbf{i}(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \cdots + i_2x^2 + i_1x + i_0.$$

Znamo da kod blok kodova imamo k informacionih bita, ali polinom je stepena $k-1$, zato što je koeficijent i_0 uz jedinični član $x^0 = 1$. Slično, kodni polinom $\mathbf{c}(x)$, koji predstavlja kodnu riječ dužine n, mora biti $n-1$ reda:

$$\mathbf{c}(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_2x^2 + c_1x + c_0.$$

Polinom $\mathbf{g}(x)$ naziva se generatorskim polinomom i očigledno mora biti reda $m = n - k$. Na primjer, kod Hemingovog koda (7,4) generatorski polinom mora biti trećeg reda, dok kod koda (15,11) on mora biti četvrtog reda. Iz razloga koji će, tokom izlaganja, biti kasnije jasniji, kod kodova koji ispravljaju jednu pogrešku generatorski polinom mora da bude prost polinom, odnosno polinom koji bez ostatka ne dijeli nijedan drugi polinom manjeg reda. Stoga se često, umjesto oznake $\mathbf{g}(x)$, koristi oznaka $\mathbf{p}(x)$ da bi se naglasilo da je u pitanju prosti polinom.

Pitanje prostosti binarnih polinoma (polinoma sa koeficijentima iz skupa {0, 1}) dosta je interesantno. Prosti polinomi prvog reda su očigledno:

$$x, x+1,$$

dok je jedini prosti polinom drugog reda:

$$x^2 + x + 1.$$

Objasnimo da ovdje, za razliku od polinoma sa koeficijentima iz skupa cijelih ili realnih brojeva, polinom $x^2 + 1$ nije prost! Naime, on je djeljiv sa $x+1$ u posmatranom binarnom polju:

$$\begin{array}{l} (x^2 + 0x + 1) : (x+1) = x+1 \\ \underline{-x^2 - x} \\ \qquad\quad x + 1. \end{array}$$

Uvijek imajte na umu da su u binarnoj algebri operacije sabiranja i oduzimanja ekvivalentne. Dakle, suprotno onome što imamo za polinome definisane nad skupom realnih i cijelih brojeva (odnosno odgovarajuće vektorske prostore), kod polinoma definisanih nad skupom binarnih brojeva (nad binarnim alfabetom) $x^2 + 1$ nije prost polinom jer je kvadrat polinoma $x+1$:

$$x^2 + 1 = (x+1)^2.$$

Provjera da li je polinom $x^2 + x + 1$ obavlja se dijeljenjem sa prostim polinomima manjeg stepena (u našem slučaju x i $x+1$). Dakle, provjera da li je neki polinom prost uvijek se može obaviti dijeljenjem toga polinoma sa prostim polinomima nižeg reda. Provjerimo za polinom $x^2 + x + 1$:

$$x^2+x+1 : x = x+1$$
$$\underline{x^2}$$
$$x+1$$
$$\underline{x}$$
$$1 \text{ (ostatak)}.$$

Na isti način, dijeljenjem ovog polinoma sa $x+1$, dobićemo rezultat x i ostatak 1.

Postoji ukupno osam polinoma trećeg reda:

$$x^3, \quad x^3+1, \quad x^3+x, \quad x^3+x+1$$
$$x^3+x^2, \quad x^3+x^2+x, \quad x^3+x^2+x+1, \quad x^3+x^2+1.$$

Svi oni polinomi koji imaju nulti član uz 1 (uz x^0) ne mogu biti prosti jer su zasigurno djeljivi sa prostim polinomom prvoga reda x. Dakle, uzimajući u obzir samo tu činjenicu, broj mogućih prostih polinoma trećeg reda svodimo na četiri:

$$x^3+1, \quad x^3+x+1 \quad x^3+x^2+x+1, \quad x^3+x^2+1.$$

Pretraga za prostim polinomima je od značaja u teoriji kodova, ali i u nekim izvedenim oblastima kao što je kriptografija. Stoga se iznalaze razni postupci kojima bi se broj pretraga smanjio što je više moguće. Mi bismo sada trebali da provjerimo djeljivost preostala četiri polinoma kandidata sa drugim prostim polinomom prvog reda $x+1$. Međutim, i ovdje imamo srećnu okolonost da se ova činjenica provjerava veoma lako. Pretpostavimo da polinom nije prost jer je djeljiv sa $x+1$, te da se može zapisati kao:

$$\mathbf{f}(x) = (x+1)\mathbf{q}(x).$$

Dakle, $x=-1$ je nula ovoga polinoma (pošto se radi o binarnoj algebri, $x=1$ je isto što i $x=-1$). Da bi $x=1$ bilo nula polinoma, to znači da u polinomnom izrazu mora da bude paran broj nenultih članova, a to dalje znači da je svaki polinom koji ima paran broj nenultih članova djeljiv sa $x+1$. Posmatrajmo dva takva među polinomima kandidatima:

$x^3+1=(x+1)(x^2+x+1)=x^3+x^2+x^2+x+x+1$ (svi zbirovi istih članova se poništavaju)

$$x^3+x^2+x+1=(x+1)(x^2+1)=(x+1)^3.$$

Ovim smo potvrdili zaključak da svi binarni polinomi sa parnim brojem nenultih članova nisu prosti. Nakon samo ove dvije opservacije, dobili smo dva prosta polinoma trećeg reda (teže se to postiže kod polinoma višeg reda):

$$x^3+x+1 \qquad x^3+x^2+1.$$

Postavlja se pitanje: kako smo utvrdili da su ovi polinomi prosti samo na osnovu činjenice da nisu djeljivi sa (prostim) polinomima prvog reda x i $x+1$? Da su djeljivi sa polinomom prvog reda, onda bi kao rezultat dijeljenja polinoma trećeg reda sa polinomom prvog reda bio polinom drugog reda, dakle polinom drugog reda bi se dobio kao količnik. Stoga se, provjerom djeljivosti sa polinomima prvog reda, u predmetnom slučaju ujedno vrši i provjera djeljivosti sa polinomom drugog reda, što predstavlja određenu uštedu. Postoje još dvije činjenice koje vrijedi pomenuti odmah na početku, a kasnije će biti dokazane u dijelu sa primjerima. Prva je da ne može biti prost binarni polinom koji ima nenulte članove samo uz parne stepene. Druga je da ako smo odredili prost polinom, možemo često na osnovu njega da jednostavno odredimo drugi prosti polinom sa revertovanim koeficijentima uz članove. Na primjer, koeficijenti prostog polinoma x^3+x+1 mogu se zapisati kao vektor $\{1, 0, 1, 1\}$, a promjenom njihovog redosljeda, od posljednjeg ka prvom, dobijamo $\{1, 1, 0, 1\}$, što su koeficijenti prostog polinoma x^3+x^2+1.

Što se tiče polinoma četvrtog reda, njih ima ukupno 16. Kada odbacimo one koji su djeljivi sa x (kojima je koeficijent uz $x^0=1$ jednak nuli), ostaje ih tačno osam. Preostali polinomi imaju član x^4+1 i potencijalno još maksimalno tri nenulta koeficijenta, a znamo da ukupan broj nenultih koeficijenata u riječi mora biti neparan, tako da preostaju samo četiri potencijalna kandidata za proste polinome:

$$x^4+x+1 \qquad x^4+x^2+1 \qquad x^4+x^3+1 \qquad x^4+x^3+x^2+x+1.$$

Kod drugog od ovih polinoma x^4+x^2+1 nenulti su samo članovi uz parne stepene, pa, kao što smo već rekli, on ne može biti prost. Preostala tri polinoma su prosti, što se može i dokazati njihovim dijeljenjem sa prostim polinomima prvog i drugog reda. Zapravo, nema ni potrebe da ih dijelimo sa polinomima prvog reda jer smo mogućnost djeljivosti sa njima već eliminisali time što nismo uzeli u obzir polinome koji nemaju slobodni član (uz stepen x^0) i one polinome koji imaju paran broj nenultih koeficijenata. Stoga je dovoljno provjeriti djeljivost polinoma sa prostim polinomom drugog reda x^2+x+1:

$$\begin{aligned}
&x^4+0x^3+0x^2+x+1 : x^2+x+1 = x^2+x\\
&\underline{x^4+x^3+x^2}\\
&\quad x^3+x^2+x+1\\
&\quad \underline{x^3+x^2+x}\\
&\qquad\qquad 1 \text{ ostatak pri dijeljenju}\\
&x^4+x^3+x^2+x+1 : x^2+x+1 = x^2+x\\
&\underline{x^4+x^3+x^2}\\
&\qquad\qquad x+1 \text{ ostatak pri dijeljenju.}
\end{aligned}$$

Dakle, vidimo da su ova dva polinoma prosti (ponekad se kaže **nesvodivi**), a čim smo provjerili prvi od njih, znamo da je i polinom x^4+x^3+1 prost jer su mu

koeficijenti {1, 1, 0, 0, 1} u reversnom redosljedu u odnosu na x^4+x+1 {1, 0, 0, 1, 1}. Napominjemo da ovo zapravo važi i kod polinoma $x^4+x^3+x^2+x+1$ koji ima sve jedinice {1, 1, 1, 1, 1}, pa je isti i kada je napisan u obrnutom redosljedu. Često se u teoriji kodova prost polinom definiše pod strožim uslovima. Pored nesvodivosti, traži se da ne dijeli nijedan polinom x^m-1 stepena koji je manji od $m<2^s-1$, gdje je s stepen nesvodivog polinoma, a dijeli bez ostatka ovaj polinom za $m=2^s-1$. U našem slučaju je $s=4$, $m=15$, a prva dva polinoma zadovoljavaju neophodnu osobinu da bi bila prosta, dok treći polinom ne zadovoljava jer dijeli bez ostatka x^5-1 (odnosno x^5+1):

$$x^5+1=(x^4+x^3+x^2+x+1)(x+1).$$

Podrobno ćemo o ovim osobinama govoriti kasnije i dodatno ukazati na njihovu važnost. Naredni bitan pojam koji uvodimo kod prostih polinoma je nula prostog polinoma. Uzmimo primjer prostog polinoma:

$$\mathbf{g}(x)=x^3+x+1.$$

Nulom prostog polinoma naziva se argument $x=a$, za koji:

$$\mathbf{g}(a)=0.$$

Očigledno je da nema skalara koji bi mogao da predstavlja nulu prostog polinoma pošto $\mathbf{g}(0)=0\cdot 0\cdot 0+0+1=1$, odnosno $\mathbf{g}(1)=1\cdot 1\cdot 1+1+1=1+1+1=1$. Kako skalar ne može biti nula ovoga polinoma, onda mora da bude vektor. Međutim, može se pokazati da vektor dužine dva ne može biti nula ovog prostog polinoma. Stoga, pretpostavimo da je nula ovog polinoma vektor kolona dužine tri:

$$a=\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}.$$

Ako je ovaj polinom nula prostog polinoma, tada mora da važi da je:

$$a^3+a+1=0.$$

Dolazimo do pitanja: šta je treći stepen nule prostog polinoma? Međutim, taj treći stepen zavisi ne samo od a već i od načina kako usvajamo nulti stepen nule prostog polinoma $a^0=1$. Podsjetimo se da je drugi prosti polinom trećeg stepena x^3+x^2+1, te da kod njega za nulu prostog polinoma važi $a^3+a^2+1=0$, odnosno da treći stepen nule prostog polinoma ovdje zavisi i od drugog stepena nule prostog polinoma.

Bez potrebe za daljom (i dubljom) analizom, možemo usvojiti sljedeća pravila: za prosti polinom r-tog stepena usvajamo stepene nule prostog polinoma a^0, a^1, ..., a^{r-1} kao nenulte, različite, linearno nezavisne vektor kolone dužine r.

Bez gubitka opštosti u slučaju izabranog prostog polinoma, odabraćemo da su tri prva stepena nule prostog polinoma:

$$a^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad a^1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad a^2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Ovo se može smatrati za zgodan izbor koji formira jediničnu matricu $[a^2 a^1 a^0]$. Isti proces može se primijeniti i na slučajeve kada baratamo polinomima višeg reda. Ipak, zapamtimo, što će u razvoju nekih narednih kodova biti korišćeno, da se mogu izabrati bilo koja tri nenulta linearno nazavisna polinoma. Na primjer, $a^0 = [0\ 0\ 1]^T$, $a^1 = [0\ 1\ 0]^T$, $a^2 = [0\ 1\ 1]^T$ nisu pogodni jer je njihova linearna kombinacija (suma) jednaka nuli: $a^0 + a^1 + a^2 = 0$. Posmatrajmo sada čemu su jednaki stepeni nule prostog polinoma višeg reda. Znamo da važi:

$$a^3 + a + 1 = 0 \Rightarrow a^3 = a + 1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$a^4 = a \cdot a^3 = a^2 + a = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \qquad a^5 = a \cdot a^4 = a^3 + a^2 = a^2 + a + 1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$a^6 = (a^3)^2 = (a+1)^2 = a^2 + 1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$a^7 = a \cdot a^6 = a(a^2 + 1) = a^3 + a = a + 1 + a = 1 = a^0.$$

Vidimo da formiramo ciklus u kome je $a^r = a^{r\%7}$, odnosno u opštem slučaju će važiti: $a^r = a^{r\%n}$, gdje je n dužina kodne riječi.

Napišimo sada stepene nule prostog polinoma (do pojave periodičnosti/cikličnosti) u obliku matrice:

$$\begin{matrix} a^6 & a^5 & a^4 & a^3 & a^2 & a^1 & a^0 \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Dobijamo kontrolnu matricu sa jednom mogućom permutacijom kolona kontrolne matrice.

Na sličan način se postupak može obaviti i za prosti polinom $\mathbf{p}(x)=x^3+x^2+1$. Započnimo sa iste prve tri kolone kao u prethodnom slučaju:

$$a^0=\begin{bmatrix}0\\0\\1\end{bmatrix} \qquad a^1=\begin{bmatrix}0\\1\\0\end{bmatrix} \qquad a^2=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

$$a^3+a^2+1=0 \Rightarrow a^3=a^2+1=\begin{bmatrix}1\\0\\0\end{bmatrix}+\begin{bmatrix}0\\0\\1\end{bmatrix}=\begin{bmatrix}1\\0\\1\end{bmatrix}$$

$$a^4=a\cdot a^3=a^3+a=a^2+a+1=\begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$a^5=a\cdot a^4=a^3+a^2+a=a^2+1+a^2+a=a+1=\begin{bmatrix}0\\1\\1\end{bmatrix}$$

$$a^6=a\cdot a^5=a(a+1)=a^2+a=\begin{bmatrix}1\\1\\0\end{bmatrix}$$

$$a^7=a\cdot a^6=a(a^2+a)=a^3+a^2=a^2+a^2+1=1=\begin{bmatrix}0\\0\\1\end{bmatrix}.$$

Matrica (kontrolna) sa različitim stepenima nule prostog polinoma je u ovom slučaju samo permutacija prethodne matrice:

$$\begin{matrix}a^6 & a^5 & a^4 & a^3 & a^2 & a^1 & a^0\end{matrix}$$
$$\begin{bmatrix}1 & 0 & 1 & 1 & 1 & 0 & 0\\ 1 & 1 & 1 & 0 & 0 & 1 & 0\\ 0 & 1 & 1 & 1 & 0 & 0 & 1\end{bmatrix}.$$

Isti postupak možemo provesti za proste polinome četvrtog stepena. Ovdje ćemo za primjer posmatrati prosti polinom četvrtog stepena $\mathbf{p}(x)=x^4+x+1$. Kod ovog prostog polinoma usvajamo prva četiri stepena nule prostog polinoma, koji su linearno nezavisne nenulte vektor kolone sa četiri elementa. Ponovo ćemo usvojiti, radi lakšeg rada, da su u pitanju stepeni koji formiraju jediničnu matricu:

$$a^3=\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\qquad a^2=\begin{bmatrix}0\\1\\0\\0\end{bmatrix}\qquad a^1=\begin{bmatrix}0\\0\\1\\0\end{bmatrix}\qquad a^0=1=\begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

Ostale nule određujemo na osnovu relacije: $\mathbf{p}(a)=a^4+a+1=0$, odnosno $a^4=a+1$. Slijedi:

$$a^4=a+1=\begin{bmatrix}0\\0\\1\\1\end{bmatrix}\quad a^5=a^2+a=\begin{bmatrix}0\\1\\1\\0\end{bmatrix}\quad a^6=a^3+a^2=\begin{bmatrix}1\\1\\0\\0\end{bmatrix}\quad a^7=a^3+a+1=\begin{bmatrix}1\\0\\1\\1\end{bmatrix}$$

$$a^8=a^2+1=\begin{bmatrix}0\\1\\0\\1\end{bmatrix}\quad a^9=a^3+a=\begin{bmatrix}1\\0\\1\\0\end{bmatrix}\quad a^{10}=a^2+a+1=\begin{bmatrix}0\\1\\1\\1\end{bmatrix}\quad a^{11}=a^3+a^2+a=\begin{bmatrix}1\\1\\1\\0\end{bmatrix}$$

$$a^{12}=a^3+a^2+a+1=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}\quad a^{13}=a^3+a^2+1=\begin{bmatrix}1\\1\\0\\1\end{bmatrix}\quad \boldsymbol{a}^{14}=a^3+1=\begin{bmatrix}1\\0\\0\\1\end{bmatrix}\quad a^{15}=1=a^0.$$

Uočite da su i ovdje, kao i kod prethodnih polinoma trećega reda, svi stepeni nule prostog polinoma višeg reda određeni putem usvojenih stepena. Formirajmo sada kontrolnu matricu, redajući stepene nule prostog polinoma od najvećeg (koji se ne ponavlja, a to je a^{n-1}) ka nižim stepenima:

$$\begin{matrix}a^{14} & a^{13} & a^{12} & a^{11} & a^{10} & a^9 & a^8 & a^7 & a^6 & a^5 & a^4 & a^3 & a^2 & a & a^0\end{matrix}$$

$$\mathbf{H}=\begin{bmatrix}1&1&1&1&0&1&0&1&1&0&0&1&0&0&0\\0&1&1&1&1&0&1&\mathbf{0}&1&1&0&0&1&0&0\\0&0&1&1&1&1&0&1&0&1&1&0&0&1&0\\1&1&1&0&1&0&1&1&0&0&1&0&0&0&1\end{bmatrix}.$$

Sličan rezultat se dobija kada upotrijebimo drugi prosti polinom $\mathbf{p}(x)=x^4+x^3+1$. Pogledajmo sada zbog čega polinom $\mathbf{p}(x)=x^4+x^3+x^2+x+1$ nije upotrebljiv za kreiranje kontrolne matrice, odnosno za Hemingov kod. Započnimo na opisani način:

$$a^3=\begin{bmatrix}1&0&0&0\end{bmatrix}^{\mathrm{T}}\qquad a^2=\begin{bmatrix}0&1&0&0\end{bmatrix}^{\mathrm{T}}$$

$$a=\begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}^{\mathrm{T}} \qquad a^0=1=\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^{\mathrm{T}}$$

$$a^4=a^3+a^2+a+1=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$$

$$a^5=a\cdot a^4=a(a^3+a^2+a+1)=$$
$$=a^4+a^3+a^2+a=a^3+a^2+a+1+a^3+a^2+a=1=a^0.$$

Dakle, $a^5=a^0$ i svi se stepeni višeg reda ponavljaju sa periodom 5 (po modulu 5). Da bi polinom bio pogodan za kreiranje Hemingovog koda, potreban uslov je da je prost, ali taj uslov nije i dovoljan, već za polinom reda r mora da važi da je najmanji stepen veći od nule, za koji važi da je $a^q=a^0$ jednak dužini kodne riječi $q=n$, što je vezano sa redom polinoma kao $n=2^r-1$. U nekim udžbenicima se samo ovakvi polinomi nazivaju prostim, dok se oni koji su prosti, ali kod kojih se dešava da je $a^q=a^0$ za $0<q<n$, nazivaju nesvodivim. Postavlja se pitanje: kakva je konstruktivna veza između informacionih bita, prostog polinoma i kodne riječi? Prvo, prosti polinom koji koristimo za generisanje koda za ispravljanje jedne pogreške u kodnoj riječi naziva se **generatorski polinom**. Kod svih kodova koje razmatramo, polinom kojim predstavljamo kodnu riječ označimo kao $\mathbf{c}(x)$. U uslovima kada nema grešaka u kodnoj riječi djeljiv je bez ostatka sa generatorskim polinomom. Uvedimo dva principa generisanja koda na osnovu generatorskog polinoma.

VI.3.1 Koder sa pomjeračkim registrom i povratnom spregom

Oba kodera koja će biti razmatrana zasnovana su na pomjeračkom registru. Pomjerački registar se sastoji od ćelija D flip-flopova, koje imaju prostu funkciju da kada bit dođe na ulaz, on se smjesti u ćeliju registra, a staro stanje registra (nula ili jedinica) prosljeđuje se na izlaz registra. Prije nego prikažemo konkretnu realizaciju, posmatrajmo četiri informaciona bita 1 0 0 1 koja ćemo prikazati u obliku informacionog polinoma:

$$\mathbf{i}(x)=x^3+0x^2+0x+1=x^3+1.$$

U kodni polinom postavimo ove informacione bite na početna mjesta u kodnoj riječi:

$$\mathbf{c}=[1\ 0\ 0\ 1\ *\ *\ *],$$

odnosno

$$\mathbf{c}(x)=x^6+x^3+c_2x^2+c_1x+c_0.$$

Kontrolni biti se moraju izabrati tako da je $\mathbf{c}(x)$ djeljivo sa generatorskim polinomom $\mathbf{g}(x) = \mathbf{p}(x) = x^3 + x + 1$. Generatorski polinom će biti označavan sa $\mathbf{g}(x)$ pošto će u nekim slučajevima biti različit od prostog polinoma. Podijelimo $\mathbf{c}(x)$ sa $\mathbf{g}(x)$:

$$\begin{array}{l} x^6 + x^3 + c_2x^2 + c_1x + c_0 : x^3 + x + 1 = x^3 + x \\ \underline{x^6 + x^4 + x^3} \\ \quad x^4 + c_2x^2 + c_1x + c_0 \\ \quad \underline{x^4 + x^2 + x} \\ \quad\quad (c_2 + 1)x^2 + (c_1 + 1)x + c_0. \end{array}$$

Dobijeni ostatak pri dijeljenju $(c_2 + 1)x^2 + (c_1 + 1)x + c_0$ mora biti 0, pa koeficijenti polinoma koji predstavljaju bite parnosti moraju da zadovolje sljedeću relaciju za date informacione bite:

$$c_2 = 1 \qquad c_1 = 1 \qquad c_0 = 0,$$

tako da je ispravna kodna riječ

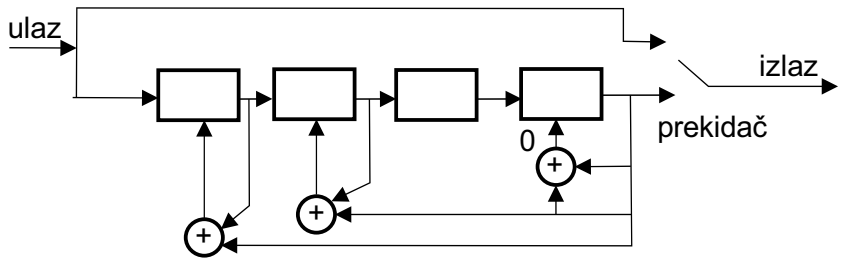
$$\mathbf{c} = [1\ 0\ 0\ 1\ 1\ 1\ 0],$$

odnosno polinom:

$$\mathbf{c}(x) = x^6 + x^3 + x^2 + x.$$

Naredno pitanje vezano je za hardversku strukturu kojom se realizuje predmetni kod, odnosno koja je u stanju da na osnovu ulaznih bita produkuje kodnu riječ. Kakav god da je hardver u pitanju, mora da bude u stanju da informacione bite neizmijenjene propusti na izlaz.

Na Slici VI.1 prikazana je struktura kodera sa povratnom spregom. U koderu postoje dvije grane. Kada je prekidač u gornjem položaju, na izlaz prevodi informacione bite. Istovremeno, informacioni biti pune pomjerački registar bez obavljanja drugih operacija. Kada je pomjerački registar popunjen, prekidač se postavlja na sredinu, čekajući da se obavi operacija proračuna kontrolnih bita u pomjeračkom registru sa povratnom spregom.



Slika VI.1. Koder sa povratnom spregom koda (7.4)

Kao i mnogo puta do sada, najbolje se poslužiti primjerom koji će ilustrovati način rada ovog kodera.

Primjer VI.3. Pretpostavimo da su informacioni biti 1001, kao i u prethodno opisanom primjeru. Proveli smo ih na izlaz i postavili u pomjerački registar u kojem je trenutno stanje 1001. Logičke operacije nisu vršene tokom punjenja registra. U ovom trenutku prekidač se nalazi u neutralnom stanju i čeka rezultat rada kodera da bi se odredili biti parnosti.

Tekuće stanje u pomjeračkom registru je:

$$1 \qquad 0 \qquad 0 \qquad 1.$$

Jedinica iz posljednje ćelije se povratnom spregom vraća prema prvoj, drugoj i četvrtoj ćeliji registra, tako da dolazi do sabiranja aktuelnog stanja u registru sa jedinicama na navedenim pozicijama, pa dobijamo:

$$1+1=0 \qquad 0+1=1 \qquad 0 \qquad 1+1=0.$$

Ovdje možemo da uočimo da koje god je stanje u prvoj ćeliji registra da će se poslije povratne sprege stanje na tom bitu dovesti na nulu, tako da je, umjesto kola koje obavlja operaciju ekskluzivno ili, dovoljno da resetujemo tu ćeliju u registru.

U narednom (drugom) koraku pomjeramo bite, uvodeći nulu u registar, čime dobijamo:

$$0 \qquad 0 \qquad 1 \qquad 0.$$

Povratna sprega sada nema dejstvo, tako da registar ostaje u istom ovom stanju. Zatim, ponovo pomjeramo stanja u registru za jedno mjesto udesno, uvođenjem nule u prvu ćeliju, čime dobijamo:

$$0 \qquad 0 \qquad 0 \qquad 1.$$

Ali sada u posljednjoj ćeliji registra imamo bit 1 koji povratno djeluje na prvu, drugu i posljednju ćeliju:

$$0+1=1 \qquad 0+1=1 \qquad 0 \qquad 1+1=0.$$

Konačno, u posljednjem koraku uvodimo novu nulu (to je ukupno treća), čime dobijamo:

$$0 \qquad 1 \qquad 1 \qquad 0.$$

Biti parnosti su oni koji su upisani u prve tri ćelije registra (kao što smo rekli, posljednji bit je zasigurno uvijek jednak 0 i ne nosi informaciju). Zatim, prekidač prelazi u položaj dolje da bi prihvatio ta tri bita parnosti u procesu tokom kojeg

se ne obavljaju operacije sa povratnom spregom, i biti parnosti izlaze redom kao 110, čime smo dobili odgovarajuću kodnu riječ. □

Za drugi koder Hemingovog (7,4) koda, koji realizuje sistem sa generatorskim polinomom koji je jednak prostom polinomu $\mathbf{g}(x) = \mathbf{p}(x) = x^3 + x^2 + 1$, koder sa povratnom spregom i pomjeračkim registrom mijenja se samo utoliko što se jedno kolo za ekskluzivno ili operaciju pomjera za jedno mjesto udesno (Slika VI.2). Da bi se lakše pamtila struktura ovakvih kodera, možete smatrati da povratna sprega na prvom bitu odgovara članu 1 u generatorskom polinomu, povratna sprega na drugom bitu nenultom članu uz koeficijent x, povratna sprega na drugom bitu je povratna sprega za nenulti koeficijent na poziciji x^2 itd.

Da bismo razumjeli ponašanje i konstrukciju kodera za slučaj kodova sa većom dužinom kodne riječi, posmatraćemo primjer koji ilustruje način kako se može kreirati koder za duže Hemingove kodove i obavljanje odgovarajućih operacija.

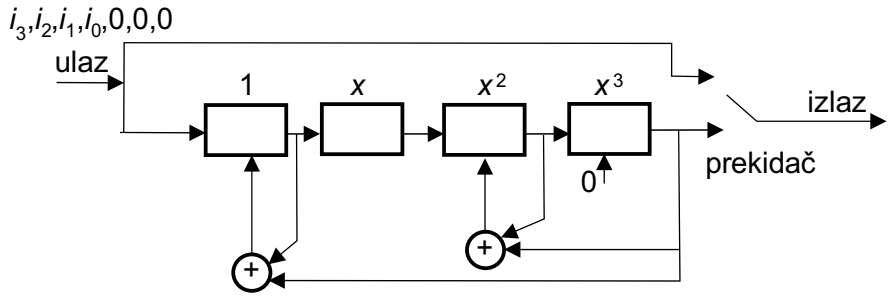
Primjer VI.4. Posmatrajmo Hemingov kod (15,11) sa generatorskim, prostim polinomom $\mathbf{g}(x) = \mathbf{p}(x) = x^4 + x + 1$. Neka je riječ koju treba kodirati: 11101000100. Izvršiti kodiranje (dopunjavanje kontrolnim bitima) ove poruke putem registra sa pomjeračkim registrom i provjerom parnosti. Prikazati navedenu hardversku strukturu.

Rješenje: Navedeni informacioni polinom može se zapisati (uzimamo da je prvi bit najveće važnosti) kao:

$$\mathbf{i}(x) = x^{10} + x^9 + x^8 + 0x^7 + x^6 + 0x^5 + 0x^4 + 0x^3 + x^2 + 0x + 0,$$

gdje su naglašeni i nulti koeficijenti radi lakšeg praćenja procedure. Korespondentni kodni polinom je:

$$\mathbf{c}(x) = x^{14} + x^{13} + x^{12} + 0x^{11} + x^{10} + 0x^9 + 0x^8 + \\ + 0x^7 + x^6 + 0x^5 + 0x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$$



Slika VI.2. Koder sa generatorskim polinomom $x^3 + x^2 + 1$

gdje su na kraj kodne riječi dodati kontrolni biti (provjere parnosti) koje treba podesiti tako da je kodni polinom djeljiv sa generatorskim polinomom. Obavimo (djelimično) postupak dijeljenja kodnog polinoma sa generatorskim polinomom:

$$x^{14}+x^{13}+x^{12}+0x^{11}+x^{10}+0x^{9}+0x^{8}+$$
$$+0x^{7}+x^{6}+0x^{5}+0x^{4}+c_3x^{3}+c_2x^{2}+c_1x+c_0 : x^{4}+x+1 = x^{10}.$$

Sada treba oduzeti od kodnog polinoma polinom:

$$x^{10}(x^{4}+x+1) = x^{14}+0x^{13}+0x^{12}+x^{11}+x^{10}.$$

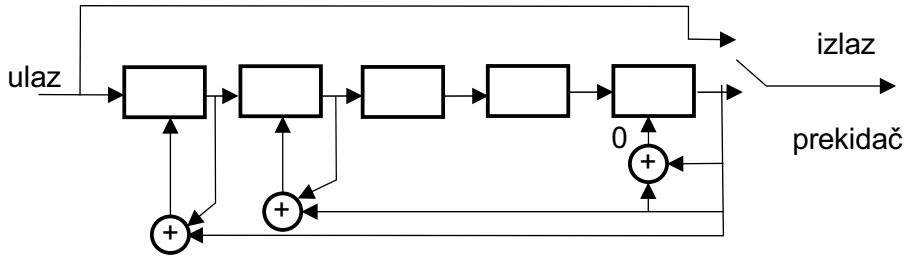
Vidimo da će samo pet koeficijenata kodnog polinoma biti promijenjeno oduzimanjem. Ovo nam govori da registar treba da ima pet ćelija. Dakle, oduzimanje u pet ćelija obavlja se na pozicijama koje korespondiraju stepenima generatorskog polinoma. Zatim pomjeramo registar za jednu poziciju da bi ušao naredni stepen (sada je to član uz x^9), te ponavljamo dalje operacije:

$$x^{13}+x^{12}+x^{11}+0x^{10}+0x^{9}+0x^{8}+$$
$$+0x^{7}+x^{6}+0x^{5}+0x^{4}+c_3x^{3}+c_2x^{2}+c_1x+c_0 : x^{4}+x+1 = x^{9}.$$

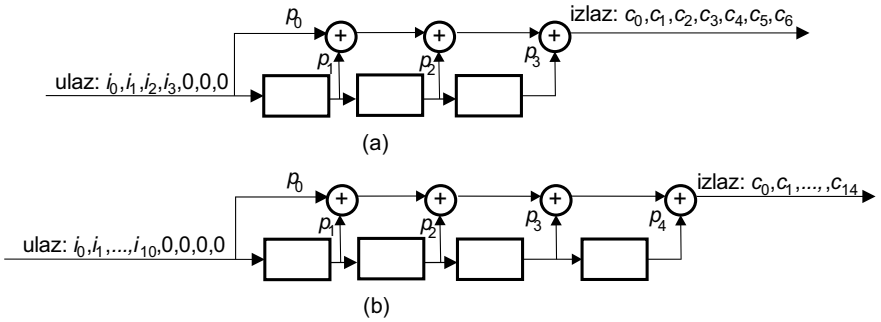
Ponovimo dijeljenje ostatka sa prostim polinomom:

$$x^{12}+x^{11}+x^{10}+x^{9}+0x^{8}+0x^{7}+x^{6}+$$
$$+0x^{5}+0x^{4}+c_3x^{3}+c_2x^{2}+c_1x+c_0 : x^{4}+x+1 = x^{8}.$$

Ako imamo da je količnik jednak nenultom koeficijentu, to znači da se u prvoj, četvrtoj i petoj ćeliji registra oduzimaju koeficijenti, odnosno da dolazi do primjene ekskluzivno ili operacije na koeficijent rezultata dijeljenja i sadržaj ćelije registra. Isti postupak se obavlja i kada imamo da je rezultat jednak nultom koeficijentu, s tim da ekskluzivno ili operacija tada neće promijeniti sadržaj ćelija registra. Nakon što se završi određivanje sadržaja registra nakon protoka (u ovom slučaju 11 informacionih bita), na zatečeno stanje se primjenjuje postupak opisan u Primjeru VI.3, kako bi se odredili biti parnosti. Na Slici VI.3 prikazan je predmetni koder.



Slika VI.3. Koder sa generatorskim polinomom x^4+x+1



Slika VI.4. Opšti slučaj kodera sa množenjem polinoma za (a) Hemingov (7,4) kod i (b) Hemingov (15,11) kod

Na sličan način realizujte koder za slučaj kada je prosti generatorski polinom $x^4 + x^3 + 1$, odnosno za proste polinome višeg reda.

VI.3.2 Koder sa pomjeračkim registrom i množenjem polinoma

Uočavamo da struktura sa pomjeračkim registrom i povratnom spregom ima manu u načinu kako se obavlja povratna sprega, sa komplikacijama koje iz tog postupka proizlaze. Stoga se, umjesto ovakvog kodera, ponekad koristi koder koji prosto treba da pomnoži informacioni polinom (u aritmetici po modulu 2) sa prostim generatorskim polinomom. Za opšti slučaj generatorskih polinoma trećeg i četvrtog reda, ova struktura je prikazana na Slici VI.4(a) i VI.4(b). Koeficijenti p_i predstavljaju koeficijente prostog generatorskog polinoma (ovdje poređani od bitova manje ka onima veće važnosti). Posmatrajmo slučaj kodera za Hemingov kod (7,4), koji ima tri ćelije u pomjeračkom registru (broj ćelija registra je jednak broju kontrolnih bita $m = n - k$). Na početku su ćelije registra prazne. U prvu ćeliju ulazi bit i_0, potiskujući nule ka preostale dvije ćelije. Na izlazu dobijamo:

$$c_0 = i_0 p_0.$$

Sljedeći bit koji ulazi u sistem je i_1, koji ulazi u prvu ćeliju registra, potiskujući i_0 u drugu ćeliju. Na izlazu dobijamo:

$$c_1 = i_0 p_1 + i_1 p_0.$$

Naredni bit je i_2, koji pomjera bite i_1 u naredne ćelije registra, produkujući izlaz:

$$c_2 = i_0 p_2 + i_1 p_1 + i_2 p_0.$$

Naredni informacioni bit i_3, pomjera i_2 u drugu i treću ćeliju registra, dok i_0 izlazi van i posljednji put se koristi za dobijanje bita kodne riječi:

$$c_3 = i_0 p_3 + i_1 p_2 + i_2 p_1 + i_3 p_0.$$

U preostala tri koraka u sistem ulaze nule, koje pored produkovanja naredna tri bita kodne riječi pripremaju koder i za kodiranje narednog bloka informacionih bita. Dobijamo:

$$c_4 = i_1 p_3 + i_2 p_2 + i_3 p_1 \qquad c_5 = i_2 p_3 + i_3 p_2 \qquad c_3 = i_3 p_3.$$

Provjerimo da li ovo korespondira sa množenjem informacionog polinoma sa generatorskim polinomom:

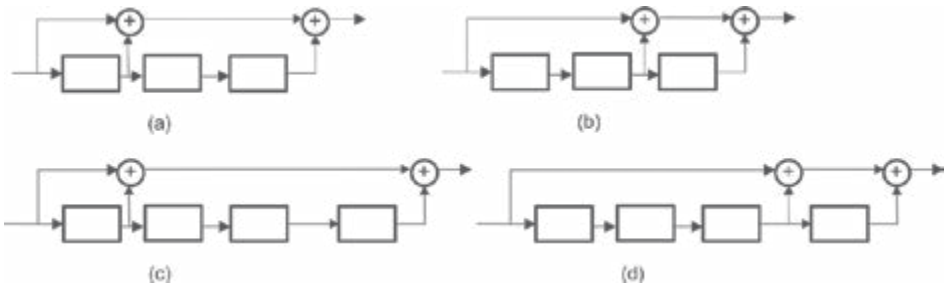
$$\begin{aligned}\mathbf{c}(x) = [i_0 + i_1 x + i_2 x^2 + i_3 x^3][p_0 + p_1 x + p_2 x^2 + p_3 x^3] = \underbrace{i_0 p_0}_{c_0} + \underbrace{i_0 p_1 + i_1 p_0}_{c_1} x + \\ + \underbrace{i_0 p_2 + i_1 p_1 + i_2 p_0}_{c_2} x^2 + \underbrace{i_0 p_3 + i_1 p_2 + i_2 p_1 + i_3 p_0}_{c_3} x^3 + \\ + \underbrace{i_1 p_3 + i_2 p_2 + i_3 p_1}_{c_4} x^4 + \underbrace{i_2 p_3 + i_3 p_2}_{c_5} x^5 + \underbrace{i_3 p_3}_{c_6} x^6.\end{aligned}$$

Jasno je da predmetni polinom generiše kolo koje je dato na Slici VI.4(a). Srećna okolnost u binarnoj aritmetici je da se prilikom množenja sa 1 ne vrši nikakava operacija, dok množenje sa 0 predstavlja izostanak neke od grana u kolu. Na Slici VI.5 prikazali smo kodere za četiri koda koja ćemo uglavnom koristiti u našim primjerima: Hemingove kodove (7,4), generisane putem prostih polinoma $x^3 + x + 1$ i $x^3 + x^2 + 1$ i Hemingove kodove (15,11), generisane putem prostih polinoma $x^4 + x + 1$ i $x^4 + x^3 + 1$.

Odredimo kodnu riječ za generatorski polinom $x^3 + x + 1$ kod koda (7,4):

$$\begin{aligned}\mathbf{c}(x) = [i_0 + i_1 x + i_2 x^2 + i_3 x^3][1 + x + x^3] = i_0 + (i_0 + i_1)x + (i_1 + i_2)x^2 + \\ + (i_0 + i_2 + i_3)x^3 + (i_1 + i_3)x^4 + i_2 x^5 + i_3 x^6.\end{aligned}$$

Vidjeli smo da je ovaj postupak suštinski jednostavniji nego kod pomjeračkog registra sa povratnom spregom, te je i hardver jednostavniji. Ta korist ne dolazi bez problema. U prethodnoj relaciji uočite da se biti i_0, i_2 i i_3 informacione riječi



Slika VI.5. Koderi sa pomjeračkim registrom i množenjem polinoma za četiri koda: (a) i (b) Hemingove kodove (7,4); (c) i (d) Hemingove kodove (15,11)

pojavljuju direktno u kodnoj riječi, dok se bit i_1 ne pojavljuje direktno, već se mora tražiti čak i kada nema greške u kodnoj riječi. Ovakav postupak ne daje **sistematski kod** (kao prethodni) gdje se informacioni biti pojavljuju direktno u kodnoj riječi, što predstavlja određenu manu ovog postupka. Da sublimiramo, suštinski imamo dva postupka za kodiranje, kod kojih se u oba slučaja koristi pomjerački registar. U jednom, sa povratnom spregom, sistem je složeniji, ali imamo sistematski kod. U drugom imamo jednostavniji hardver i postupak, ali gubimo sistematičnost koda.

VI.3.3 Dekodiranje Hemingovih kodova

Dekodiranje Hemingovih kodova koji ostvaruju jednu korekciju greške može se obaviti dijeljenjem primljene riječi sa generatorskim polinomom. U ovom materijalu ne ulazimo u sami postupak dijeljenja, odnosno u hardversku strukturu sistema za dijeljenje, ali ako ne postoji bolji postupak, suštinski se može koristiti struktura sa povratnom spregom, kao što je to rađeno kod prvog tipa kodera (sa pomjeračkim registrom i povratnom spregom).

Jasno je da možemo smatrati da nema pogreške u kodu ako dobijemo polinom količnik bez ostatka. Međutim, mnogo nam je bitnije šta se dešava u uslovima kada se pojave pogreške. Pretpostavimo da je poslata riječ predstavljena kodnim polinomom $\mathbf{c}(x)$, a da je primljena predstavljena polinomom $\mathbf{r}(x)$ koji se na jednoj poziciji razlikuje od poslatog kodnog polinoma:

$$\mathbf{r}(x) = \mathbf{c}(x) + x^l = \mathbf{i}(x)\mathbf{g}(x) + x^l.$$

Kada podijelimo polinom $\mathbf{r}(x)$ sa generatorskim polinomom, dobijamo ostatak pri dijeljenju koji je jednak ostatku pri dijeljenju x^l sa $\mathbf{g}(x)$. Uočite da ako je x^l manjeg stepena od $\mathbf{g}(x)$, ostatak je jednak x^l, dok u ostalim slučajevima imamo rezultate koji korespondiraju $x^l \% \mathbf{g}(x)$ i nisu jednaki x^l. Naš zadatak je da, na osnovu tako dobijenog ostatka, odredimo l, odnosno poziciju na kojoj se pogreška dogodila.

Dakle, da rezimiramo: ako je polinom ostatak $\mathbf{rem}(x) = \mathbf{r}(x) \% \mathbf{g}(x)$ jednak stepenu x^l, greška se dogodila na poziciji koja korespondira stepenu l. Ako to nije slučaj, možemo uzastopnim množenjem sa x^{-1} da dobijemo:

$$\mathbf{rem}(x) \cdot \underbrace{x^{-1} \cdot x^{-1} \cdots x^{-1}}_{p-\text{puta}} = x^q.$$

Zaključujemo da se greška dogodila na poziciji koja korespondira sa x^{p+q}, odnosno na poziciji stepena $p+q$. Prije nego na jednom primjeru ilustrujemo predmetnu aritmetiku, ostaje da se protumači šta je to stepen x^{-1}. Vidjeli smo da je kod pogodnih generatorskih polinoma blok kodova najniži stepen, za koji važi da je nula prostog polinoma $a^r = 1$, zapravo n. Dakle, $a^n = 1$ ili $x^n = 1$ iz našeg ugla posmatranja. Pomnožimo lijevu i desnu stranu ovog izraza sa x^{-1}, pa dobijamo:

$$x^{-1} = x^{n-1}.$$

Primjer VI.5. Neka je dat kodni polinom $\mathbf{c}(x)=x^6+x^3+x^2+x$ iz prethodnih primjera, dobijen putem kodera sa pomjeračkim registrom i povratnom spregom sa generatorskim polinomom $\mathbf{g}(x)=x^3+x+1$. Neka je zbog greške u kanalu primljena riječ:

$$\mathbf{r}(x)=x^6+x^5+x^3+x^2+x.$$

Dekodirati poslatu poruku.

Rješenje: Podijelimo primljenu riječ sa generatorskim polinomom:

$$\begin{array}{l}\mathbf{r}(x):\mathbf{g}(x)=x^6+x^5+x^3+x^2+x:x^3+x+1=x^3+x^2+x+1\\ \quad\underline{x^6+x^4+x^3}\\ \qquad x^5+x^4+x^2+x\\ \qquad\underline{x^5+x^3+x^2}\\ \quad x^4+x^3+x\\ \quad\underline{x^4+x^2+x}\\ \qquad x^3+x^2\\ \qquad\underline{x^3+x+1}\\ \qquad\quad x^2+x+1.\end{array}$$

Dobijeni ostatak pri dijeljenju je $\mathbf{rem}(x)=x^2+x+1$. Sada ga pomnožimo nekoliko puta sa $x^{-1}=x^{n-1}=x^6=x^2+1$:

$$\mathbf{rem}(x)\cdot x^{-1}=(x^2+x+1)\cdot(x^2+1)=x^4+x^3+x^2+x^2+x+1=x^4+x^3+x+1.$$

Za nulu ovog prostog polinoma $x=a$ važi $x^3=x+1$, dok je $x^4=x^2+x$, pa dobijamo:

$$\mathbf{rem}(x)\cdot x^{-1}=x^2+x.$$

Moramo da odradimo još množenja sa x^{-1}:

$$\mathbf{rem}(x)\cdot x^{-1}\cdot x^{-1}=(x^2+x)\cdot(x^2+1)=x^4+x^3+x^2+x=x^2+x+x+1+x^2+x=x+1.$$

Mogli bismo, zapravo, već da prekinemo dalju proceduru pošto smo na desnoj strani dobili $x+1$ koje je jednako x^3, ali možemo i da nastavimo kao:

$$\mathbf{rem}(x)\cdot x^{-1}\cdot x^{-1}\cdot x^{-1}=(x+1)\cdot(x^2+1)=x^3+x^2+x+1=x+1+x^2+x+1=x^2.$$

Dakle,

$$\mathbf{rem}(x)=x^2\cdot x^3=x^5.$$

Ako vas buni predmetna algebra, uvijek možete imati na umu da su ovo iste operacije koje se mogu provoditi sa stepenima nule prostog polinoma. Dakle, sada smo detektovali da se greška dogodila na poziciji uz x^5, pa možemo korigovati primljenu riječ kao:

$$\mathbf{c'}(x)=\mathbf{r}(x)+x^5=x^6+x^3+x^2+x.$$

Kako su kod predmetnog koda informacioni biti postavljeni na početku kodne riječi (kod je sistematski), znamo da je poslata poruka [1 0 0 1]. □

Slično se postupa i kod koda koji je dobijen putem množenja informacionog polinoma sa generatorskim polinomom, samo što on nema sistematičnost, pa se moramo još malo potruditi oko određivanja informacionog polinoma. Uočimo takođe da je ostatak jednak tačno koloni kontrolne matrice, koja se može kreirati od stepena nule prostog polinoma, na poziciji gdje se dogodila pogreška (kolona $[1\ 1\ 1]^T$). Da li se ta informacija može koristiti za pojednostavljivanje postupka? Samo onda kada nam je kontrolna matrica poznata, a mi je, zbog potrebe da se vrši pretraga, te zbog potrebe da se memoriše, često u praksi izbjegavamo. Ponekad se radi i tako što se zapamti nekoliko stepena nule prostog polinoma (nekoliko kolona matrice), pa se množenje sa x^{-1} provodi dok se ne dođe do neke od memorisanih nula prostog polinoma. Ipak, riječ je o detaljima koji su od važnosti za tehničku realizaciju kodnih postupaka, pa ih dalje nećemo razmatrati.

VI.3.4 Cikličnost Hemingovih kodova

Definicija VI.1. Kod se naziva **cikličnim** ako se cikličnom rotacijom ulijevo i udesno svake ispravne kodne riječi $\mathbf{c}(x)$ dobija ispravna kodna riječ. □

Ciklična rotacija kodne riječi ulijevo (ili udesno) podrazumijeva da se biti pomjeraju ulijevo (ili udesno) i da se oni koji „ispadnu“ ponovo javljaju na kraju (ili početku) kodne riječi. Na primjer, ako je kodna riječ:

$$1\ 0\ 0\ 1\ 1\ 1\ 0.$$

Ciklično pomjeranje za dva mjesta ulijevo znači:

$$0\ 1\ 1\ 1\ 0\ \underline{1}\ \underline{0},$$

gdje su podvučeno označeni karakteri koji su „ispali“ s početka riječi i premješteni na njen kraj. Slično, za pomjeranje udesno za tri mjesta imamo:

$$\underline{1}\ \underline{1}\ \underline{0}\ 1\ 0\ 0\ 1\ 1.$$

Uočimo bitnu činjenicu da je ciklično pomjeranje za $n-1$ mjesta u jednu stranu ekvivalentno cikličnom pomjeranju za jedno mjesto u drugu stranu.

Hemingovi kodovi, dobijeni putem prostih polinoma na opisani način, su ciklični.

Primjer VI.6. Dokazati da je Hemingov kod, formiran putem generatorskog polinoma $\mathbf{p}(x) = \mathbf{g}(x) = x^3 + x + 1$, cikličan.

Rješenje: Ako osobina cikličnosti važi za pomjeranje za jedno mjesto ulijevo, onda ona važi za svako pomjeranje, jer se dalje može primijeniti za pomjeranje pomjerene riječi. Ujedno, ciklično pomjeranje udesno takođe se može realizovati

pomjeranjima ulijevo. Dakle, svodimo dokazivanje predmetne činjenice na to da ako je $\mathbf{c}(x)$ ispravna kodna riječ, onda je to i riječ nastala iz predmetne riječi cikličnim pomjeranjem ulijevo za jednu poziciju. Ako je kodni polinom sa koeficijentom 0 uz član x^6, tada pomjeranje ulijevo za 1 zapravo predstavlja množenje tog polinoma sa x. Ako je polazni polinom djeljiv sa prostim polinomom, tada je i dobijeni polinom djeljiv sa njime (samo je količnik pomnožen sa x). Situacija se malo komplikuje ako je koeficijent uz najveći član 1. Tada polazni polinom, za koji pretpostavljamo da je prost, možemo zapisati kao:

$$\frac{x^6+\mathbf{r}(x)}{x^3+x+1}=x^3+\mathbf{q}(x),$$

gdje je polinom $\mathbf{p}(x)$ reda koji je manji ili jednak 5, polinom $x^3+\mathbf{q}(x)$ je količnik, dok je $\mathbf{q}(x)$ polinom reda ne većeg od 2. Ako ovaj polinom pomjerimo ulijevo za 1, dobijamo polinom $\mathbf{r}(x)x+1$ za koji treba dokazati da je djeljiv prostim polinomom $\mathbf{g}(x)$. Iz prve relacije jednostavno slijedi:

$$\mathbf{r}(x)=x^4+x^3+\mathbf{q}(x)(x^3+x+1),$$

pa se problem svodi na dokaz da je polinom:

$$x^5+x^4+x\mathbf{q}(x)(x^3+x+1)+1$$

djeljiv sa (x^3+x+1), što se ponovo svodi na to da je potrebno dokazati da je polinom x^5+x^4+1 djeljiv sa x^3+x+1. Količnik ova dva polinoma je x^2+x+1:

$$x^5+x^4+1=(x^3+x+1)(x^2+x+1)$$

bez ostatka, pa smo dokazali predmetnu tvrdnju za kodove koji su generisani putem predmetnog prostog polinoma. □

U pitanju je netrivijalna procedura koja se može obaviti za sve do sada uvedene kodove. Postoji i drugi sistematičniji način koji se može provesti kod svih kodova. Naime, ako je prvi bit u kodnoj riječi jednak 1, polinom se može zapisati kao:

$$x^{n-1}+\mathbf{r}(x),$$

gdje je $\mathbf{q}(x)$ polinom reda ne većeg od $n-2$. Pomnožimo ovaj polinom sa x i dobijamo:

$$x^n+x\mathbf{r}(x).$$

Pod pretpostavkom da je polazni polinom djeljiv sa prostim generatorskim polinomom, onda je i ovaj polinom djeljiv sa prostim generatorskim polinomom. Međutim, mi umjesto ovog polinoma imamo polinom $x\mathbf{r}(x)+1$ koji se od osnovnog polinoma, koji je djeljiv sa prostim polinomom, razlikuje u tome što je oduzeto x^n i dodato 1. Zaključujemo da ako se želi održati djeljivost sa prostim polinomom,

koji je korišćen kao generator koda, mora da važi da je $x^n - 1$ djeljivo sa tim prostim polinomom. U algebri sa binarnim alfabetom ovo znači da $x^n + 1$ (operacije sabiranja i oduzimanja su ekvivalentne kod binarnog alfabeta) mora biti djeljivo sa prostim polinomom.

Kod nebinarnih alfabeta, što ćemo vidjeti u narednoj sekciji, važi prethodno izloženo pravilo da $(x^n - 1)$ mora biti djeljivo s prostim polinomom koji se koristi za generisanje koda, a operacija oduzimanja mora biti u skladu s posmatranim nebinarnim alfabetom.

VI.4 Nebinarni kodovi

VI.4.1 Kontrolna i generišuća matrica

Prije nego uvedemo kontrolnu matricu za nebinarne Hemingove kodove, podsjetimo se kontrolne matrice Hemingovog binarnog (7,4) koda. Neka, polazeći od posljednje ka prvim kolonama, svaka kolona predstavlja binarno zapisan broj, počevši od 1 do 7:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Podsjetimo se da su kolone matrice morale da budu nenulte i različite da bi mogle da omoguće dekodiranje putem sindroma. Dekodiranje ovog koda obavlja se množenjem primljene kodne riječi sa (transponovanom) kontrolnom matricom:

$$\mathbf{S} = \mathbf{c}\mathbf{H}^{\mathrm{T}}.$$

U slučaju da dobijemo da je sindrom **S** jednak nekoj od kolona, onda se greška pojavila na toj lokaciji. Ako dobijemo sindrom sa svim nulama (ova kombinacija se ne pojavljuje u kontrolnoj matrici), podrazumijevamo da nema greške u kodnoj riječi. Napominjemo da je množenje obavljano kao logička i operacija (mada isti smisao ima i standardno matematičko množenje), dok je sabiranje obavljano preko ex-ili operacije koja se može smatrati i operacijom „po modulu 2“, odnosno od dobijenog rezultata se odredi ostatak pri dijeljenju sa 2. Alternativno, kontrolnu matricu možemo definisati tako što prvo postavimo jediničnu matricu na kraj kontrolne matrice, a ispred nje ređamo sve kolone koje se razlikuju od prethodno napisanih kolona:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Ako je kontrolna matrica zapisana u ovom obliku: $\mathbf{H} = [\mathbf{P} \mid \mathbf{I}_{n-k}]$, tada se generišuća matrica, kojom množimo informacione bite da bismo dobili kodnu riječ, može zapisati kao:

$$\mathbf{G} = \left[\mathbf{I}_k \mid \mathbf{P}^{\mathrm{T}}\right].$$

Nebinarni kodovi su obično definisani u aritmetici po modulu nekog većeg prostog broja. Npr. kod ternarnog koda, kodni simboli se usvajaju kao 0, 1, 2, pa se operacije množenja i sabiranja obavljaju u aritmetici po modulu 3:

$$\begin{array}{llllll} 0\cdot 0=0 & 0\cdot 1=0 & 0\cdot 2=0 & 0+0=0 & 0+1=1 & 0+2=2 \\ 1\cdot 0=0 & 1\cdot 1=1 & 1\cdot 2=2 & 1+0=1 & 1+1=2 & 1+2=0 \\ 2\cdot 0=0 & 2\cdot 1=2 & 2\cdot 2=1 & 2+0=2 & 2+1=0 & 2+2=1. \end{array}$$

Prije nego damo matematički formalniju definiciju nebinarnog Hemingovog koda, posmatrajmo ternarni Hemingov kod (4,2). Neka je njegova kontrolna matrica data kao:

$$\mathbf{H} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

Uočava se da smo postavili da su informacioni simboli posljednji u kodnoj riječi. Neka je dobijena kodna riječ $\mathbf{c} = [1\ 2\ 2\ 0]$. Sindrom koji dobijamo u ovom slučaju je:

$$\mathbf{c}\mathbf{H}^{\mathrm{T}} = \begin{bmatrix} 1 & 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [0\ \ 0].$$

Dakle, u pitanju je ispravna kodna riječ jer je dobijen sindrom koji je jednak nula vektoru. Neka se sada dogodila greška na prvom bitu i neka je prvi bit jednak 2. Dobijeni sindrom je:

$$\begin{bmatrix} 2 & 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [2\ \ 1].$$

Sindrom nam kaže da se greška dogodila na prvom simbolu jer je sadržaj sindroma jednak prvoj koloni kontrolne matrice. Za sada sve funkcioniše po logici kao kod binarnih kodova. Posmatrajmo sljedeću situaciju, sa prvim simbolom primljene riječi 0:

$$\begin{bmatrix}0 & 2 & 2 & 0\end{bmatrix}\begin{bmatrix}2 & 1\\ 1 & 1\\ 1 & 0\\ 0 & 1\end{bmatrix}=[1\ \ 2].$$

Dobili smo kolonu koja ne postoji u kontrolnoj matrici **H**, ali i dalje identifikuje da se pogreška pojavljuje na prvoj poziciji, jer je dobijeni sindrom jednak dvostrukoj vrijednosti prve kolone. Naime,

$$2[2\ \ 1]=[4\ \ 2]=[1\ \ 2].$$

Dakle, sada ovo možemo da zapišemo nešto opštije ako je kodna riječ **c** i ako joj je dodat vektor pogreške **e** koji ima jedan nenulti elemenat. Dobijeni sindrom je:

$$\mathbf{r}\mathbf{H}^{\mathrm{T}}=(\mathbf{c}+\mathbf{e})\mathbf{H}^{\mathrm{T}}=\mathbf{c}\mathbf{H}^{\mathrm{T}}+\mathbf{e}\mathbf{H}^{\mathrm{T}}=\mathbf{e}\mathbf{H}^{\mathrm{T}}.$$

Ako je težina pogreške jednaka 1, onda je sindrom jednak koloni matrice **H** i dekodiranje poruke se obavlja tako što se od kodne riječi **x** na odgovarajućoj poziciji oduzme 1, dok ako je jednak umnošku kolone matrice **H**, onda se taj umnožak oduzima od odgovarajuće pozicije. U našem slučaju (slučaju ternarnog koda), oduzimanje 1 je ekvivalentno sabiranju sa 2, odnosno oduzimanje 2 je ekvivalentno sabiranju sa 1.

Sad možemo i da razumijemo način na koji se dobija kontrolna matrica (a ne da je dajemo ad hok). Jasno je da se u kontrolnoj matrici ne može dva puta pojaviti ista kolona, ali ni njen umnožak. Pretpostavimo da imamo dva informaciona simbola. Polazna matrica ima dvije vrste. Ostaje da se ispita koliko nam treba kolona za provjere „parnosti“:

$$\begin{bmatrix}? & 1 & 0\\ ? & 0 & 1\end{bmatrix}.$$

Očigledno se sada u kontrolnoj matrici ne mogu pojaviti kolone $[2\ 0]^{\mathrm{T}}$ i $[0\ 2]^{\mathrm{T}}$ jer su to umnošci postojećih kolona. Prva naredna moguća kombinacija je $[1\ 1]^{\mathrm{T}}$, koja eliminiše kombinaciju $[2\ 2]^{\mathrm{T}}$. Konačno, preostaje još samo kombinacija $[1\ 2]^{\mathrm{T}}$, koja eliminiše kombinaciju $[2\ 1]^{\mathrm{T}}$. Dobijena kontrolna matrica je:

$$\mathbf{H}=\begin{bmatrix}1 & 1 & 1 & 0\\ 2 & 1 & 0 & 1\end{bmatrix}.$$

Ništa ne ometa da Hemingov ternarni kod (4,2) ima kontrolnu matricu:

$$\mathbf{H}=\begin{bmatrix}2 & 1 & 2 & 0\\ 1 & 1 & 0 & 1\end{bmatrix}.$$

Jednostavno rečeno, kod ternarnog koda postoje dva konkurenta za svaku kolonu kontrolne matrice, a ne treba zaboraviti ni to da, kad se dobije kontrolna matrica, možemo izmiješati (permutovati) njene kolone na bilo koji pogodan način.

Postavlja se pitanje: koje dimenzije ovakvog koda moraju biti? Pretpostavimo da imamo r-arni Hemingov kod sa k informacionih simbola (sada ne možemo govoriti o bitovima). Broj kontrolnih bita je u ovom slučaju $n-k$, a toliko je i vrsta kontrolne matrice. Postavlja se pitanje: kolika nam dužina kodne riječi treba da bismo izvršili jedno ispravljanje pogreške? Očigledno je, po prethodnom izlaganju, da svaka kolona matrice eliminiše $r-2$ mogućih kolona matrice koje su umnošci date kolone sa brojevima 2, ..., $r-1$ (množenje sa 1 daje samu tu kolonu, dok množenje sa 0 daje kolonu koja je neupotrebljiva kod korekcije pogreški). Sa r simbola može se kreirati ukupno r^{n-k} različitih kolona u matrici koja ima $n-k$ vrsta. Jedna kolona sa svim nulama je neupotrebljiva, pa stoga, prilikom konstrukcije kontrolne matrice, mi možemo koristiti $r^{n-k}-1$ kolona. Kada odaberemo jednu kolonu, mi eliminišemo još $r-2$ kolona. Stoga je maksimalan broj kolona matrice (odnosno, broj bita kontrolnih + informacionih) jednak:

$$n=\frac{r^{n-k}-1}{r-1}.$$

Provjera na poznatim slučajevima koda sa $r=2$ i recimo $n-k=3$ daje $n=7$ (Hemingov kod (7,4)), dok, recimo, $r=3$ i $n-k=2$ daje $n=4$ (ternarni Hemingov kod (4,2) koji je korišten kao primjer). Napominjemo da u gornjoj relaciji može da stoji znak $\leq$. Naime, za datu redundanciju $n-k$ možemo da imamo maksimalno k informacionih simbola ili maksimalno n ukupnih simbola. Kodovi koji zadovoljavaju ovu relaciju sa jednakošću nazivaju se perfektnim. Sada se postavlja pitanje: kako, recimo, na osnovu poznatog k i r doći do n? To je, zapravo, problem koji nas interesuje, jer obično znamo koliko informacionih bita želimo da šaljemo (to je k), kao i koji alfabet koristimo za kodiranje (to je r). Postoje različiti pristupi, ali se uglavnom svode na malo numerike.

Naredni problem sa kojim se moramo suočiti je da li postoji način da se ovakav tip koda generiše na osnovu odgovarajuće generatorske matrice $\mathbf{G}$. Generatorska matrica mora da ima dimenzije $k\times n$ i mora da zadovoljava da je $\mathbf{G}\mathbf{H}^{\mathrm{T}}=\mathbf{0}$. Po analogiji sa binarnim kodovima, generatorsku i kontrolnu matricu možemo zapisati kao:

$$\mathbf{G}=[\mathbf{I}_k\,|\,\mathbf{R}] \qquad\qquad \mathbf{H}=[\mathbf{P}^{\mathrm{T}}\,|\,\mathbf{I}],$$

gdje smo u generatorsku matricu uveli novu matricu $\mathbf{R}$ (napominjemo da kod binarnih kodova na ovom mjestu imamo matricu $\mathbf{P}$). Jasno je da mora da važi:

$$\mathbf{G}\mathbf{H}^{\mathrm{T}}=\mathbf{P}+\mathbf{R}=\mathbf{0}.$$

Dakle, u aritmetici sa r-arnim kodom $\mathbf{R} = -\mathbf{P}$, odnosno ovo se može zapisati (pomalo u MATLAB notaciji) kao $\mathbf{R} = r - \mathbf{P}$. Npr. za posmatrani ternarni kod to se svede na:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 2 & 2 \end{bmatrix}$$

$$\mathbf{G}\mathbf{H}^{\mathrm{T}} = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

VI.4.2 Definicija preko polinoma

Nebinarni Hemingovi kodovi se mogu uvesti i preko prostih polinoma:

$$\sum_{i=0}^{Q} a_i x^i$$

čiji koeficijenti uzimaju vrijednosti iz skupa $a_i \in \{0,1,...,r-1\}$. Stepen ovih polinoma odgovara broju provjera „parnosti“ (r-arnosti). Radi jednostavnosti, posmatrajmo ternarni kod sa 2 provjere parnosti. Postavlja se pitanje: koji su prosti polinomi reda koji je manji ili jednak sa Q? Prosti polinomi prvog reda su: x, $x+1$, $x+2$, ali, recimo, nije $2x+1$ jer podijeljen sa 2 daje $x+2$, za koji smo usvojili da je prost. Po istoj logici možemo reći da polinom $x+2$ nije prost jer $2(2x+1) = x+2$. Da bismo pojednostavili razmatranje, usvojimo da prost polinom mora imati koeficijent uz član najvišeg reda jednak 1 (ovakvi polinomi se nazivaju **moničnі**) i ne provjeravajmo dijeljenja sa konstantom. Broj mogućih r-arnih moničnih je r^Q. U slučaju ternarnog koda ti polinomi su:

x^2 (nije prost jer je djeljiv sa x)

x^2+1 (prost! što nije kod binarnog alfabeta)

x^2+2 (nije prost, djeljiv je sa $x+1$ i $x+2$)

x^2+x (nije prost jer je djeljiv sa x i $x+1$)

x^2+x+1 (nije prost $=(x+2)^2$)

x^2+x+2 (prost)

x^2+2x (nije prost $=x(x+2)$)

x^2+2x+1 (nije prost $=(x+1)^2$)

x^2+2x+2 (prost).

Detektovali smo (prostom pretragom) 3 prosta polinoma (preciznije nesvodiva) traženog oblika. Međutim, nisu svi prosti polinomi pogodni za kodiranje Hemingovog koda. Podsjetimo se da je kod binarnih kodova bilo potrebno da stepeni nule

prostog polinoma generatora koda a^l budu jednaki $a^0 = 1$ za $l = n$, gdje je n dužina kodne riječi, ali ne i prije $l = n$. Stoga otpada polinom $x^2 + 1$ jer je s njime djeljiv $x^l + 1$ za $l = 2 < n$. Preostaju samo dva prosta polinoma koja se mogu upotrijebiti za datu namjenu; oba zadovoljavaju osobinu da ni $x^2 + 1$ ni $x^3 + 1$ nije djeljivo sa njima bez ostatka. Polinom $x^4 + 1$ je djeljiv, što ćemo pokazati na primjeru polinoma $x^2 + x + 2$ (isto važi i za polinom $x^2 + 2x + 2$):

$$
\begin{array}{lll}
x^4+1 & : \quad x^2+x+2= & x^2 \\
-(x^4+x^3+2x^2) & & \\
\underline{-x^3-2x^2+1=2x^3+x^2+1} & : \quad x^2+x+2= & 2x \\
-(2x^3+2x^2+4x) & & \\
\underline{-x^2-4x+1=2x^2+2x+1} & : \quad x^2+x+2= & 2 \\
-(2x^2+2x+4) & & \\
1-4=-3=0. & &
\end{array}
$$

Dakle: $x^4 + 1 : x^2 + x + 2 = x^2 + 2x + 2$ i to bez ostatka (ujedno smo pokazali i da je dati polinom djeljiv i sa $x^2 + 2x + 2$. Povežimo sada dobijene kontrolne matrice sa polinomom. Neka je a nula prostog polinoma. U kontrolnoj matrici upisujemo stepene nule prostog polinoma, usvajajući da je:

$$a^0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad\qquad a^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Svi ostali stepeni se mogu opisati preko ova dva stepena:

$$a^2 = -a - 2 = 2a + 1 = 2\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$a^3 = 2a^2 + a = 2\begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

Stoga smo dobili kontrolnu matricu:

$$\mathbf{H} = \begin{array}{c} \begin{matrix} a^3 & a^2 & a^1 & a^0 \end{matrix} \\ \begin{bmatrix} 2 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix} \end{array}.$$

Posmatrajmo sada sljedeći stepen:

$$a^4 = a^3 \cdot a = (2a + 2)a = 2a^2 + 2a = 4a + 2 + 2a = 2 = 2a^0.$$

Ovdje sada vidimo da kod nebinarnih kodova ne važi $a^n = a^0$ kao kod binarnih kodova, već je ovdje situacija nešto složenija, odnosno važi: $a^n = 2a^0 = (r-1)a^0 = -a^0$. U opštem slučaju, kod nebinarnih kodova važi ova relacija, dok ona važi i kod

binarnih kodova, jer je kod binarnih kodova 1 i -1 isti broj, odnosno sabiranje i oduzimanje su ista operacija.

Kodiranje preko polinoma obavlja se množenjem informacionog polinoma sa generatorskim polinomom koda. Neka je informaciona riječ u našem slučaju [2 1]. Ona se može prikazati preko polinoma kao $\mathbf{i}(x) = 2x + 1$. Kodna riječ je sada:

$$\mathbf{c}(x) = \mathbf{i}(x)\mathbf{g}(x) = (2x+1)(x^2+x+2) = 2x^3+3x^2+5x+2 = 2x^3+0x^2+2x+2.$$

Dobijena kodna riječ je [2 0 2 2]. Dekodiranje putem polinoma može se obaviti dijeljenjem sa generatorskim polinomom. Ako je dobijen rezultat bez ostatka, tada je dobijeni polinom količnik informacija koja je bila poslata. U slučaju da dobijemo ostatak pri dijeljenju, moramo odrediti kojem stepenu korijena prostog generatorskog polinoma odgovara dobijeni ostatak. Na primjer, neka je primljeni polinom: $2x^3 + 2x^2 + 2x + 2$. Obavimo proceduru njegovog dijeljenja sa generatorskim polinomom:

$$\begin{array}{lll} 2x^3+2x^2+2x+2 & : & x^2+x+2 \quad = \quad 2x \\ -(2x^3+2x^2+4x) & & \\ x+2. & & \end{array}$$

Znači, dobijeni ostatak je $x + 2$. Ovo odgovara vektor koloni $[1\ 2]^T$. Iz kontrolne matrice mi lako uočavamo da je u pitanju umnožak druge kolone matrice (koja odgovara stepenu prostog polinoma a^2) $[1\ 2]^T = 2[2\ 1]^T$, na osnovu čega zaključujemo da se pogreška dogodila na drugoj poziciji i da je težina greške 2. U praksi se do ovog zaključka ne dolazi tako lako (govorimo o slučaju dužih kodnih riječi) zato što se, zbog dimenzija, kontrolna i generatorska matrica izbjegavaju. Umjesto toga se polinom ostatak množi sa a^{-1}, što je u datom polju ekvivalentno množenju sa $-a^{n-1}$. U našem slučaju je $a^3 = 2a^2 + a = 4a + 2 + a = 2a + 2$, pa je $-a^3 = a + 1$. Pomnožimo ovaj polinom s našim ostatkom $(a+2)(a+1) = a^2 + 2 = (2a+1) + 2 = 2a$. Odavde zaključujemo da se greška dogodila na a^2 stepenu, a da je težina pogreške 2, odnosno da treba oduzeti dva na poziciji x^2 u polinomu.

VI.5 Dekodiranje više pogreški

Kod koda s dvostrukom korekcijom greške moramo poći od Hemingovih kodova sa većim brojem bita. Posmatrajmo slučaj sa 15 bita. Za sada posmatrajmo uvođenje 4 provjere parnosti. Kao jedan potencijalni polinom može poslužiti $x^4 + x + 1$. Kada odredimo stepene nule prostog polinoma, u ovom slučaju dobijamo kontrolnu matricu oblika:

$$\begin{matrix} a^{14} & a^{13} & a^{12} & a^{11} & a^{10} & a^{9} & a^{8} & a^{7} & a^{6} & a^{5} & a^{4} & a^{3} & a^{2} & a^{1} & a^{0} \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Ako se želi korigovati više grešaka, mora se dodati još nekoliko kolona matrice. Dodajmo četiri nova reda da bismo eliminisali još jednu pogrešku. Pretpostavimo da će nam za ispravljanje te dodatne pogreške biti potrebno isto onoliko bita koliko je bilo potrebno i za prvu grešku. Kasnije ćemo vidjeti nešto sistematičniji pristup u pogledu određivanja potrebnog broja dodatnih provjera parnosti (dodatnog broja redova kontrolne matrice). U dodatna četiri reda mogu da se pojave samo kolone koje su se pojavljivale i u dosadašnjoj matrici, jer se tu već sada nalaze sve moguće nenulte kombinacije od četiri bita (nulta kombinacija nam ne daje informaciju o poziciji pogreške, pa je kao takva beskorisna). Dakle, u dodatna četiri reda ponovo se moraju nalaziti stepeni nule prostog polinoma, odnosno još preciznije – dodatna četiri reda predstavljaju neku funkcionalnu zavisnost prethodna četiri reda. Označimo ovu funkcionalnu zavisnost kao $f(a)$. Ako posmatramo kolonu a^{12}, onda je u njenom nastavku $f(a^{12})$. Kada u primljenoj riječi nema pogreški, sindrom će biti nula, dok ako imamo jednu pogrešku, sindrom će biti oblika:

$$\mathbf{S} = \begin{bmatrix} a^k \\ \hline f(a^k) \end{bmatrix},$$

odnosno biće jednak koloni kontrolne matrice, kao što smo imali i do sada slučaj. Ako imamo dvije pogreške, sindrom će tada biti jednak zbiru kolona kontrolne matrice (koristimo paradigmu dekodiranja putem kontrolne matrice zbog potreba da objasnimo postupak dekodiranja pogreški i izazove koji se u njemu javljaju, a naravno da je postupak zasnovan na polinomima danas mnogo prihvatljiviji):

$$\mathbf{S} = \begin{bmatrix} a^k \\ \hline f(a^k) \end{bmatrix} + \begin{bmatrix} a^m \\ \hline f(a^m) \end{bmatrix} = \begin{bmatrix} \mathbf{S}_1 \\ \hline \mathbf{S}_2 \end{bmatrix}.$$

Ono što želimo postići je da se na osnovu dvije jednačine:

$$\mathbf{S}_1 = a^k + a^m$$
$$\mathbf{S}_2 = f(a^k) + f(a^m)$$

odrede k i m, odnosno pozicije gdje su se greške dogodile. U binarnoj aritmetici je znanje o pozicijama pogreške dovoljno za dekodiranje.

Sada je potrebno odgovoriti na pitanje o prirodi funkcije $f(a)$. Linearna funkcijska zavisnost nije pogodna i nećemo se na njoj zadržavati. Pokušajmo sa kvadratnom funkcijom, odnosno sa $f(a)=a^2$. U tom slučaju imamo sistem sa dvije jednačine i dvije nepoznate:

$$\mathbf{S}_1 = a^k + a^m$$
$$\mathbf{S}_2 = a^{2k} + a^{2m}.$$

Međutim, kvadriranjem lijeve i desne strane prve jednačine dobijamo:

$$\mathbf{S}_1^2 = a^{2k} + 2a^k a^m + a^{2m} = a^{2k} + a^{2m} = \mathbf{S}_2$$

zbog toga što u predmetnoj algebri važi $2r = r + r = 0$. Stoga moramo tražiti rješenje u nelinearnostima višeg reda. Pokazuje se da su neparne stepene funkcije pogodne za predmetnu primjenu. Za dekodiranje dvije pogreške važi $f(a)=a^3$, što daje sistem:

$$\mathbf{S}_1 = a^k + a^m$$
$$\mathbf{S}_2 = a^{3k} + a^{3m}.$$

Drugi sindrom $\mathbf{S}_2$ možemo izraziti kao:

$$\mathbf{S}_2 = a^{3k} + a^{3m} = (a^k + a^m)(a^{2k} - a^k a^m + a^{2m}) = \mathbf{S}_1(a^{2k} + a^{2m} + a^{k+m}) = \mathbf{S}_1(\mathbf{S}_1^2 + a^{k+m}).$$

U izvođenju je upotrijebljen zbir kubova, a zatim smo upotrijebili do sada izvedene činjenice vezane za prvi sindrom $\mathbf{S}_1$ i činjenicu da je sabiranje i oduzimanje u posmatranom binarnom polju ista operacija. Dakle, važi:

$$a^{k+m} = \frac{\mathbf{S}_2}{\mathbf{S}_1} - \mathbf{S}_1^2 = \frac{\mathbf{S}_2}{\mathbf{S}_1} + \mathbf{S}_1^2.$$

Formirajmo sada kontrolnu matricu ovog koda (uočimo da su donje četiri vrste jednake kubovima korespondentnih gornjih vrsta):

$$\mathbf{H}' = \left[\begin{array}{cccccccccccccc}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 1 \\
\hline
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & \mathbf{0} & 0 & 1 & 1 & 0 & 0 & 0 & 1
\end{array}\right].$$

Dekodiranje jedne pogreške se obavlja na isti način kako je do sada rađeno, a u narednom primjeru ćemo demonstrirati način dekodiranja dvije pogreške.

Primjer VI.7. Dekodirati slučaj dvije pogreške koje su se dogodile na pozicijama koje korespondiraju stepenima $k=14$ i $m=5$.

Rješenje: Sindrom je u ovom slučaju:

$$\begin{aligned}\mathbf{S} &= \left[1 \quad 0 \quad 0 \quad 1 \,\vdots\, 1 \quad 1 \quad 1 \quad 1\right]^{\mathrm{T}} + \left[0 \quad 1 \quad 1 \quad 0 \,\vdots\, 0 \quad 0 \quad 0 \quad 1\right]^{\mathrm{T}} \\ &= \left[1 \quad 1 \quad 1 \quad 1 \,\vdots\, 1 \quad 1 \quad 1 \quad 0\right]^{\mathrm{T}} = \left[a^{12} \,\vdots\, a^{11}\right]^{\mathrm{T}}.\end{aligned}$$

Dakle, dobili smo da je:

$$\mathbf{S}_1 = a^k + a^m = a^{12} \qquad\qquad \mathbf{S}_2 = a^{11}.$$

Iz druge relacije slijedi:

$$\begin{aligned}a^{k+m} &= \frac{\mathbf{S}_2}{\mathbf{S}_1} + \mathbf{S}_1^2 = \frac{a^{11}}{a^{12}} + a^{24} = a^{-1} + a^9 = a^{14} + a^9 = \\ &= [1\,0\,0\,1]^{\mathrm{T}} + [1\,0\,1\,0]^{\mathrm{T}} = [0\,0\,1\,1]^{\mathrm{T}} = a^4.\end{aligned}$$

Sada možemo (realizacija se može obaviti softverskim putem programskog ciklusa) pretpostaviti vrijednost k i na osnovu predmetne relacije odrediti m i provjeriti da li je rezultat ispravan, uvrštavanjem k i m u relaciju za sindrom $\mathbf{S}_1$.

Za $k=0$ dobijamo da je $k+m=4$, odnosno da je $m=4$. Uvrštavanjem u relaciju, za sindrom $\mathbf{S}_1$ dobijamo:

$$a^0 + a^4 = [0\,0\,0\,1]^{\mathrm{T}} + [0\,0\,1\,1]^{\mathrm{T}} = [0\,0\,1\,0]^{\mathrm{T}} = a^2 \neq a^{12} = \mathbf{S}_1.$$

Naredna vrijednost je $k=1$, pa dobijamo da je $m=3$. Uvrštavanjem u relaciju, za sindrom $\mathbf{S}_1$ dobijamo:

$$a^1 + a^4 = [0\,0\,1\,0]^{\mathrm{T}} + [0\,0\,1\,1]^{\mathrm{T}} = [0\,0\,1\,1]^{\mathrm{T}} = a^0 \neq a^{12} = \mathbf{S}_1.$$

Sljedeći pokušaj za $k=2$ bi podrazumijevao da je $m=2$ što je kontradikcija, pa se i ne provjerava. Inkrementiranjem k dobijamo $k=3$ i $m=1$, što je već provjereno za $k=1$ i $m=3$, baš kao i naredni inkrement $k=4$ i $m=0$. Za $k=5$ potrebno je da je $m=14$ da bi $k+m$ po modulu n bilo jednako 4:

$$5+14 \bmod 15 = 4.$$

Provjerimo ovu situaciju:

$$a^5 + a^{14} = [0\,1\,1\,0]^{\mathrm{T}} + [1\,0\,0\,1]^{\mathrm{T}} = [1\,1\,1\,1]^{\mathrm{T}} = a^{12} = \mathbf{S}_1.$$

Na ovaj način smo ispravno identifikovali da su se pogreške dogodile na pozicijama $k = 5$ i $m = 14$, pa smo u stanju da ih ispravimo.

Posmatrajmo sada još složeniji primjer sa ispravljanjem tri pogreške. Po prethodno izloženoj logici, kontrolnoj matrici dodajemo nove vrste koje moraju predstavljati neke permutacije kolona osnovne kontrolne matrice za ispravljanje jedne pogreške. Odnosno, dodajemo četiri nove vrste u matricu, a ponovo su, po prethodno izloženoj logici koju sada nećemo pokazivati, ove dodatne vrste sljedeći neparni (u ovom slučaju peti) stepeni kolona osnovne matrice:

$$\mathbf{H}'' = \left[\begin{array}{cccccccccccccc}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & \mathbf{0} & 1 & 0 & 0 & 0 & 1 \\
\hdashline
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & \mathbf{0} & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
\hdashline
\cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} \\
1 & 0 & 1 & \mathbf{1} & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1
\end{array}\right].$$

Uočavamo da je prva od dodatih vrsta sa svim nulama. Provjera parnosti je u ovom slučaju uvijek trivijalno zadovoljena, pa se takve vrste mogu izostaviti (kolona je prekrižena). Druga činjenica (takođe vizuelno naznačena) je da su druga i treća dodata vrsta iste. Ovo znači da bi u svim slučajevima produkovale isti rezultat prilikom provjere parnosti, pa se ovakvo ponavljanje izostavlja. Dakle, kod koji je u stanju da produkuje tri korekcije greške sa 15 bita je (15,5) jer preostaje 10 vrsta kontrolne matrice, odnosno 10 provjera parnosti.

Brisanje nepotrebnih vrsta kontrolne matrice je za nas dobitak jer smanjujemo broj provjera parnosti. Međutim, nije kod svih kodova jednostavno uočiti, kao u prethodnom primjeru, koje su vrste redundantne, odnosno koje su provjere parnosti nepotrebne. Redundantna vrsta ne mora biti nula-vrsta niti kopija prethodne vrste, već može biti linearna kombinacija prethodnih vrsta.

Kako rastu dimenzije kodne riječi, te kako se povećava potreba za ispravljanjem grešaka, opisani postupak počinje da bude nesistematičan i složen. Više problema javlja se u postupku, a ovdje pobrajamo neke od njih: problem sistematičnog

određivanja dimenzija koda (zbog redundantnih vrsta u kontrolnoj matrici), problem generisanja koda (koji i dalje nije previše složen, ali je ipak komplikovaniji nego kod Hemingovih kodova koji ispravljaju jednu pogrešku), i najznačajniji problem određivanja pozicija pogreške. Vidjeli smo da je kod ispravljanja dvije pogreške potrebno izvršiti pretragu po jednom parametru, što nije problematično, ali kod ispravljanja tri pogreške potrebna je pretraga po dvije vrijednosti stepena nule prostog polinoma, a za više pogreški i pretraga po većem broju parametara. Riječ je o značajnom usporavanju procesa dekodiranja koje sprečava korišćenje Hemingovih kodova, uvedenih na opisani „naivni" način, u brojnim praktičnim primjenama. Stoga je bio nepohodan razvoj sistematičnih postupaka za kodiranje i dekodiranje koji su u stanju da isprave veći broj pogreški.

VI.6 BCH kodovi

U klasi blok kodova **BCH kodovi** su poznata potklasa postupaka za korekciju većeg broja pogreški. Uvedeni su nezavisno od francuskog matematičara Aleksija Hokunghema (fr. *Alexis Hocquenghema*) i indijskih matematičara i statističara Radža Bozea (engl. *Raj Bose*) i Reja Čaudhurija (engl. *Dwijendra Ray-Chaudhuri*), krajem 1950-ih godina. Uz Rid – Solomonove kodove, koji će rudimentarno biti objašnjeni u Poglavlju VIII, ovo su najpoznatiji postupci za blok kodiranje koje je u stanju da ispravi veći broj pogreški. Slobodno se može reći da je u pitanju kodni postupak koji predstavlja algebarskom kodnom teorijom potkrijepljeno proširenje Hemingovih kodova. Objasnićemo proces kodiranja (jednostavan) i proces dekodiranja (komplikovan) kroz naredne podsekcije.

VI.6.1 Kodiranje BCH kodova

Bili smo u stanju da kodnu riječ Hemingovog koda zapišemo putem polinoma u obliku $\mathbf{c}(x) = \mathbf{i}(x)\mathbf{g}(x)$, gdje je generatorski polinom $\mathbf{g}(x)$ bio prost polinom u odgovarajućem polju. Vidjeli smo da se kontrolna matrica Hemingovog koda koji može da ispravi više pogreški sastoji od nekoliko podmatrica, od kojih svaka naredna omogućava po još jednu ispravku pogreške. Prvi dio ove kontrolne matrice korespondira kontrolnoj matrici Hemingovog koda. Svaki naredni povećava redundanciju, odnosno daje nove kontrolne bite, a smanjuje broj informacionih bita. Kako prva podmatrica predstavlja kontrolnu matricu Hemingovog koda, logično je da se u generatorskom polinomu BCH koda, odnosno koda koji ispravlja više pogreški, pojavljuje generatorski polinom Hemingovog koda. Stoga možemo zapisati generatorski polinom BCH koda, kao:

$$\mathbf{g}(x) = \mathbf{p}(x)\mathbf{p}_3(x)\mathbf{p}_5(x)\cdots\mathbf{p}_{2w-1}(x) = \mathbf{p}(x)\sum_{i=2}^{w}\mathbf{p}_{2i-1}(x),$$

gdje je w broj grešaka koje je kod u stanju da ispravi, dok su $\mathbf{p}_{2i-1}(x)$ polinomi koji korespondiraju dodatim kolonama u kontrolnoj matrici. Tako, na primjer, polinom $\mathbf{p}_3(x)$ bi trebao da korespondira proširenju kontrolne matrice za četiri kolone u Primjeru VI.7 za kod sa $n = 15$ bita, dok bi kod $\mathbf{p}_5(x)$ trebao da odgovara proširenju kontrolne matrice za dodatna dva reda. Stepen ovih polinoma bi morao da bude jednak broju vrsta kontrolne matrice. Stoga je prvi izazov, sa kojim se srećemo, da odredimo stepen ovih polinoma. Prije nego objasnimo postupak za određivanje stepena predmetnih polinoma, uočimo da ako je a nula prostog polinoma $\mathbf{p}(x)$ da bi nula polinoma $\mathbf{p}_3(x)$ morala biti a^3 jer ovaj polinom treba da korespondira kolonama polazne kontrolne matrice koje su stepenovane na kub. Slično, polinom $\mathbf{p}_5(x)$ bi morao da se anulira za stepen a^5: $\mathbf{p}_5(a^5) = 0$. Navodimo ove činjenice pošto će biti korišćene u nekim djelovima našeg dokaza dekodirajućeg algoritma za BCH kod premda u literaturi postoje pristupi gdje se ove činjenice mnogo eksplicitnije uvode u proces konstrukcije koda nego što je urađeno u ovom udžbeniku. Kao što već uočavamo, a do kraja udžbenika će to postati mnogo jasnije, riječ je o složenoj materiji, pa se u teoriji i literaturi koriste različiti pristupi da se uvedu i objasne pojedine pojave.

Za određivanje stepena polinoma $\mathbf{p}_{2i-1}(x)$ koristićemo koncept **konjugata**. Ovaj koncept će biti mnogo jasniji kada budemo objasnili teorijske osnove procesa dekodiranja BCH koda. Za polinom $\mathbf{p}(x)$ prvi konjugat je nula prostog polinoma a, naredni je kvadrat te nule a^2, zatim kvadrat prethodnog konjugata $(a^2)^2 = a^4$ (zaključujemo da se konjugati računaju kao kvadrati prethodnih stepena), naredni konjugat je $(a^4)^2 = a^8$. Ovaj polinom nema drugih konjugata, odnosno $(a^8)^2 = a^{16} = a^{16\%n} = a^{16\%15} = a$. Dakle, konjugati ovog polinoma su $\{a, a^2, a^4, a^8\}$. Stoga, možemo zaključiti da je u pitanju polinom četvrtog stepena jer imamo četiri različita konjugata u datom polju. Još interesantniji detalj je da se predmetni polinom može zapisati kao proizvod polinoma prvog reda čije su nule konjugati:

$$\mathbf{p}(x) = (x+a)(x+a^2)(x+a^4)(x+a^8) = x^4 + (a^8 + a^4 + a^2 + a)x^3 + (a^{12} + a^{10} + a^9 + a^6 + a^5 + a^3)x^2 + (a^{14} + a^{13} + a^{11} + \mathrm{a}^7)x + a^{15} =$$

$$= x^4 + \left(\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right) x^3 + \left(\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) x^2 +$$

$$+ \left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right) x + 1 = x^4 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} x^3 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + x^2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} x + 1 = x^4 + x + 1.$$

Uočimo da su nule prostog polinoma u ovom slučaju i a^2, a^4, a^8, odnosno konjugati. Kao što se vidi, riječ je o moćnom konceptu koji se često koristi u ovoj oblasti, ali ćemo ga mi koristiti samo povremeno. Konjugati polinoma $\mathbf{p}_3(x)$ su u datom polju (odnosno za dato n) različiti kvadratni stepeni prethodnih stepena kuba nule prostog polinoma a^3: a^3, $(a^3)^2 = a^6$, $(a^6)^2 = a^{12}$, $(a^{12})^2 = a^{24} = a^{24\%15} = a^9$. Naredni stepen bi bio $(a^9)^2 = a^{18\%15} = a^3$, čime zaključujemo da postoje ponovo četiri različita konjugata. Dakle, polinom $\mathbf{p}_3(x)$ je ponovo četvrtog stepena, sa nulama koje korespondiraju konjugatima. Za vježbu možete na isti način, na koji smo putem nula (konjugata) odredili polinom $\mathbf{p}(x)$, odrediti $\mathbf{p}_3(x)$. Ovdje će biti upotrijebljena alternativna strategija. Zaključili smo da je polinom $\mathbf{p}_3(x)$ četvrtog stepena, pa ga možemo zapisati kao:

$$\mathbf{p}_3(x) = b_{34}x^4 + b_{33}x^3 + b_{32}x^2 + b_{31}x + b_{30},$$

gdje su b_{3i}, $i = 0,1,...,4$ koeficijenti koje treba odrediti. Da bismo pojasnili izazove koji se u ovom postupku mogu desiti, nećemo usvojiti razumnu pretpostavku da je $b_{34} = 1$ (u pitanju je polinom četvrtog reda, pa koeficijent uz četvrti stepen mora da bude nenulti). Rekli smo da je nula ovoga polinoma $x = a^3$, pa mora da važi:

$$\mathbf{p}_3(a^3) = b_{34}a^{12} + b_{33}a^9 + b_{32}a^6 + b_{31}a^3 + b_{30} =$$
$$= \begin{bmatrix}1\\1\\1\\1\end{bmatrix} b_{34} + \begin{bmatrix}1\\0\\1\\0\end{bmatrix} b_{33} + \begin{bmatrix}1\\1\\0\\0\end{bmatrix} b_{32} + \begin{bmatrix}1\\0\\0\\0\end{bmatrix} b_{31} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix} b_{30} = \begin{bmatrix}0\\0\\0\\0\end{bmatrix}.$$

Suštinski, riječ je o četiri jednačine sa pet nepoznatih. Uzimajući npr. da je $b_{30} = 0$, iz posljednjeg reda (jednačine) slijedilo bi $b_{34} + b_{30} = 0$, odnosno $b_{34} = 0$, iz treće jednačine bi slijedilo da je $b_{33} = 0$, iz druge da je $b_{32} = 0$ i konačno bismo dobili da je i $b_{31} = 0$. Očigledno je da se radi o pogrešnom rezultatu jer nije u pitanju polinom odgovarajućeg reda, pa se može usvojiti da je $b_{30} = 1$, odakle slijedi da je $b_{34} = 1$, zatim $b_{33} = b_{32} = b_{31} = 1$:

$$\mathbf{p}_3(x) = x^4 + x^3 + x^2 + x + 1.$$

Konjugati a^5 za kod dužine $n = 15$ su: a^5, $(a^5)^2 = a^{10}$, dok je $(a^{10})^2 = a^{20} = a^{20\%15} = a^5$. Dakle, imamo samo dva različita konjugata, pa je polinom $\mathbf{p}_5(x)$ drugog reda:

$$\mathbf{p}_5(x) = b_{52}x^2 + b_{51}x + b_{50}.$$

Iz $\mathbf{p}_5(a^5) = 0$ slijedi:

$$\mathbf{p}_5(a^5) = b_{52}a^{10} + b_{51}a^5 + b_{50} = b_{52}\begin{bmatrix}0\\1\\1\\1\end{bmatrix} + b_{51}\begin{bmatrix}0\\1\\1\\0\end{bmatrix} + b_{50}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\end{bmatrix}.$$

Ovdje postoje, zapravo, samo dvije nezavisne jednačine jer je prvi red trivijalno zadovoljen sa $0=0$, a druga i treća jednačina su iste. Kod binarnih polinoma, pošto znamo da je u pitanju polinom drugog stepena, možemo usvojiti $b_{52}=1$, pa iz druge jednačine $b_{52}+b_{51}=0$ slijedi $b_{51}=1$, dok iz posljednje $b_{52}+b_{50}=0$ imamo $b_{50}=1$. Dakle, imamo polinom:

$$\mathbf{p}_5(x) = x^2 + x + 1.$$

Radi jasnoće treba napomenuti da se kod dužih kodnih riječi ne dešava često da su svi koeficijenti polinoma $\mathbf{p}_{2i-1}(x)$ jednaki 1, kao u našem slučaju. Stoga, predmetnu proceduru treba provoditi s pažnjom.

Očigledno se kod kodiranja BCH koda ne dešava ništa spektakularno. Odredimo generatorski polinom $\mathbf{g}(x)$ i zatim množenjem s informacionim polinomom $\mathbf{i}(x)$ odredimo kodni polinom $\mathbf{c}(x)$:

$$\mathbf{c}(x) = \mathbf{g}(x)\mathbf{i}(x).$$

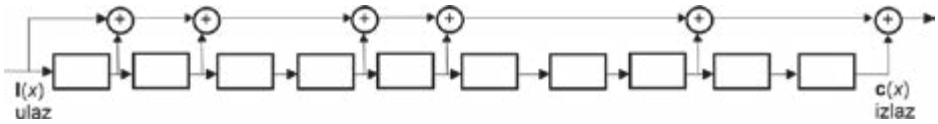
Kod se može realizovati i putem kodera sa povratnom spregom, ali i putem množenja polinoma. Množenjem polinoma ne dobijamo sistematski kod jer neće svi biti informacionog polinoma biti direktno predstavljeni u kodnoj riječi. BCH kod je blok kod. Dimenzije ovog koda se mogu zapisati kao:

$$\left(n, n - \sum_{i=1}^{w} \deg[\mathbf{p}_{2i-1}(x)]\right),$$

gdje je deg[] stepen pojedinih polinoma, s tim da smo usvojili da je $\mathbf{p}_1(x) = \mathbf{p}(x)$. U praksi se odomaćila predstava da se BCH kodovi označavaju kao BCH(n,w) ili samo (n,w), odnosno preko dužine kodne riječi i broja grešaka koje kod ispravlja, umjesto uobičajenog načina kod blok kodova (n,k) (dužina kodne riječi, broj informacionih bita). Možemo usvojiti da je BCH(n,1) kod koji ispravlja jednu pogrešku zapravo korespondentni Hemingov kod. U našem slučaju generatorski polinomi kodova BCH(15,2) i BCH(15,3) za prosti polinom $\mathbf{p}(x) = x^4 + x + 1$ su:

BCH(15,2)
$$\begin{aligned}\mathbf{g}(x) &= \mathbf{p}(x)\mathbf{p}_3(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) = \\ &= x^8 + x^7 + x^6 + x^4 + 1\end{aligned}$$

BCH(15,3)
$$\begin{aligned}\mathbf{g}(x) &= \mathbf{p}(x)\mathbf{p}_3(x)\mathbf{p}_5(x) = (x^8 + x^7 + x^6 + x^4 + 1)(x^2 + x + 1) = \\ &= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.\end{aligned}$$



Slika VI.6. Koder BCH koda (15,3), zasnovanog na prostom polinomu x^4+x+1

Naravno, mnogo su interesantniji BCH kodovi veće dužine, ali atraktivnost dužine $n=15$ potiče iz činjenice da je moguće sve ključne efekte kodiranja i dekodiranja demonstrirati, pa i sračunati, za ovako kratak kod. Na Slici VI.6 prikazan je koder BCH(15,3) zasnovan na prostom polinomu x^4+x+1. Uočavamo da se, osim povećanja dužine registra, koji odgovara redu polinoma, koder ne razlikuje od kodera koji realizuju Hemingov kod množenjem polinoma.

Množenjem informacionog polinoma sa generatorskim polinomom, za kod (15,3) dobijamo:

$$\begin{aligned}\mathbf{c}(x) &= (i_0+i_1x+i_2x^2+i_3x^3+i_4x^4)(x^{10}+x^8+x^5+x^4+x^2+x+1) = \\ &= i_0+(i_0+i_1)x+(i_0+i_1+i_2)x^2+(i_1+i_2+i_3)x^3+(i_0+i_2+i_3+i_4)x^4+ \\ &\quad +(i_0+i_1+i_3+i_4)x^5+(i_1+i_2+i_4)x^6+(i_2+i_3)x^7+(i_0+i_3+i_4)x^8+(i_1+i_4)x^9+ \\ &\quad +(i_0+i_2)x^{10}+(i_1+i_3)x^{11}+(i_2+i_4)x^{12}+i_3x^{13}+i_4x^{14}.\end{aligned}$$

Uočavamo da se u kodnoj riječi direktno pojavljuju informacioni biti i_0, i_3 i i_4, dok se ostali nalaze „smiješani“. Međutim, u uslovima kada nema pogrešaka, dekodiranje je izuzetno jednostavno. U sljedećoj sekciji govorimo o znatno složenijem izazovu – dekodiranju BCH kodova.

VI.6.2 Dekodiranje BCH kodova – teorijski osnovi

U Poglavlju I uveli smo čuveni Euklidski algoritam za određivanje najvećeg zajedničkog djelioca. Pored osnovne forme i rekurzivne formulacije, posmatrali smo ekstenziju ovog algoritma na polinome, te na polinome definisane preko konačnih polja. Posebno smo se osvrnuli na mogućnost da se najveći zajednički djelilac dva broja (ili izraza) a i b napiše u formi linearne kombinacije ta dva broja:

$$\gcd = u\,a+v\,b.$$

Objasnili smo i iterativni postupak kojim se određuju koeficijenti a i b.

Algoritam inicijalizujemo sa $r_{-1}=a$, $r_0=b$, $u_{-1}=1$, $u_0=0$, $v_{-1}=0$ i $v_0=1$.

Za svaki naredni korak k računamo količnik q_{k+1} i ostatak pri dijeljenju r_{k+1} za r_{k-1} i r_k:

$$r_{k-1}=q_{k+1}r_k+r_{k+1},$$

nakon čega se računaju:

$$u_{k+1} = u_{k-1} - q_{k+1}u_k \qquad\qquad v_{k+1} = v_{k-1} - q_{k+1}v_k,$$

tako da u svakom koraku važi linearna kombinacija $r_k = u_k\, a + v_k\, b$. Proces se nastavljao sve dok ne dobijemo da je r_{k-1} djeljivo sa r_k i u tom koraku usvajamo odgovarajuće v_k i u_k kao rezultate algoritma, odnosno kao težine u linearnoj kombinaciji.

Kodnu riječ BCH koda možemo zapisati kao:

$$\mathbf{c}(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} = \sum_{i=0}^{n-1} c_i x^i .$$

Znamo da su nule polinoma $\mathbf{p}_{2i-1}(x)$, $i = 1, 2, \ldots, w$, ujedno i nule kodne riječi jer su polinomi faktori kodne riječi s informacionim polinomom. Dakle, nule a, a^3, ..., a^{2w-1} moraju biti nule kodnog polinoma. Lako je pokazati da su nule kodnog polinoma svi stepeni $\{a^i, i = 1,\ldots,2w\}$, dakle ne samo neparni stepeni već i „susjedni" parni stepeni. Kvadrirajmo kodni polinom:

$$\mathbf{c}^2(x) = \left(\sum_{i=0}^{n-1} c_i x^i \right)^2 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} c_i x^i c_j x^j = \sum_{i=0}^{n-1} c_i^2 x^{2i} + 2\sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} c_i x^i c_j x^j .$$

Iz dvostruke sume izdvojili smo podsumu za koju važi $i = j$. Drugi izraz je nula jer treba imati na umu da je u pitanju sabiranje po modulu 2, a to znači da bilo koji binarni broj sabran sa samim sobom daje nulu. Prva suma u gornjem izrazu je, zapravo, polazni polinom, ali kada se umjesto x uvrsti x^2. Dakle, važi:

$$\mathbf{c}^2(x) = \mathbf{c}(x^2).$$

Svaka nula polinoma $\mathbf{c}(x)$ je nula i njegovog kvadratnog stepena, a to dalje znači da ako je a nula kodnog polinoma, njegova nula mora biti i a^2. Slično se može pokazati i za druge nule. Uvrštavanjem bilo kojeg od prvih $2w$ stepena nule prostog polinoma u kodni polinom, dobijamo nulu:

$$\mathbf{c}(a^j) = \sum_{i=0}^{n-1} c_i \left(a^j\right)^i = 0 \qquad j = 1, 2, \ldots, 2w.$$

Svako odstupanje od ovakvog rezultata je sindrom koji ukazuje da nešto nije u redu sa primljenom kodnom riječju, odnosno da u njoj postoje greške. Svi stepeni nule prostog polinoma predstavljaju kolone kontrolne matrice odgovarajućeg Hemingovog koda. Promjenom redosljeda kolona Hemingovog koda ne dolazi do promjene u karakteristikama koda. Stoga se stepeni a^i mogu zamijeniti kolonama Hemingove matrice, koje možemo indeksirati kao a_i. Na osnovu toga važi:

$$\sum_{i=0}^{n-1} c_i a_i^j = 0 \qquad j = 1, 2, \ldots, 2w.$$

Vrijedi napomenuti da su BCH kodovi ciklični, što se može dokazati na isti način kako je to urađeno kod Hemingovog koda. To znači da bilo koje ciklično pomjeranje kodne riječi daje kodnu riječ BCH koda. Ovo nam omogućava da možemo zapisati:

$$\sum_{i=0}^{n-1} c_i a_i^{-j} = 0 \quad j = 1,2,...,2w,$$

što suštinski označava samo promjenu poretka u kontrolnoj matrici koda. Obje ove pretpostavke uvedene su samo radi nešto jednostavnijih izvođenja u nastavku. Ipak, treba ih imati na umu u procesu dekodiranja, što se posebno odnosi na drugu činjenicu.

U slučaju pojave greški primljena riječ se razlikuje od kodne riječi:

$$\mathbf{r}(x) = \mathbf{c}(x) + \mathbf{e}(x)$$
$$r_i = c_i + e_i .$$

Broj pogreški kod BCH koda ograničen je na w i zavisi od dizajna koda, odnosno kodovi se dizajniraju da isprave do w pogreški. Uvrštavanjem $\mathbf{r}(x)$ (odnosno r_i) umjesto $\mathbf{c}(x)$ (odnosno c_i) na prijemu (na prijemu želimo da odredimo $\mathbf{c}(x)$ na osnovu $\mathbf{r}(x)$ koje nam je poznato), prethodni izraz postaje:

$$S_X = \sum_{i=0}^{n-1} r_i a_i^{-j} = \sum_{i=0}^{n-1} c_i a_i^{-j} + \sum_{i=0}^{n-1} e_i a_i^{-j} = \sum_{i=0}^{n-1} e_i a_i^{-j} \quad j = 1,2,...,2w.$$

S_j nazivamo sindromskim koeficijentima koji zavise samo od pozicija gdje su se pogreške dogodile, odnosno ovi koeficijenti ukazuju na pozicije gdje je kod „bolestan“. Na osnovu sindromskih koeficijenata kreira se sindromski polinom:

$$\mathbf{S}(x) = \sum_{j=1}^{2w} S_j x^{j-1} .$$

Dakle, naš cilj je da na osnovu sindromskog polinoma, koji zavisi isključivo od pozicija gdje su se pogreške dogodile u kodnoj riječi, odredimo pozicije i korigujemo riječ i ispravno je dekodiramo. Uvrštavajući izraze za S_j u sindromski polinom, dobijamo:

$$\mathbf{S}(x) = \sum_{j=1}^{2w} S_j x^{j-1} = \sum_{j=1}^{2w} x^{j-1} \sum_{i=0}^{n-1} r_i a_i^{-j} = \sum_{i=0}^{n-1} r_i \sum_{j=1}^{2w} x^{j-1} a_i^{-j}$$
$$= \sum_{i=0}^{n-1} r_i a_i^{-1} \frac{1 - x^{2w} a_i^{-2w}}{1 - x a_i^{-1}} = \sum_{i=0}^{n-1} r_i \frac{x^{2w} a_i^{-2w} - 1}{x - a_i}.$$

Uvrštavajući $r_i = c_i + e_i$, dobijamo:

$$\mathbf{S}(x)=\sum_{i=0}^{n-1} r_i \frac{x^{2w}a_i^{-2w}-1}{x-a_i}=\sum_{i=0}^{n-1} c_i \frac{x^{2w}a_i^{-2w}-1}{x-a_i}+\sum_{i=0}^{n-1} e_i \frac{x^{2w}a_i^{-2w}-1}{x-a_i}=\sum_{i=0}^{n-1} e_i \frac{x^{2w}a_i^{-2w}-1}{x-a_i},$$

jer je sindrom za ispravnu kodnu riječ (suma sa članovima c_i) jednak nuli. Dakle, sindrom ne zavisi od same kodne riječi, već samo od (pozicija ili vektora) pogreški e_i. Uočimo dalje da je samo manje ili jednako od w članova prethodne sume različito od nule jer maksimalno w ima pogreški u riječi (nenultih koeficijenata polinoma $\mathbf{e}(x)$). Neka se pogreška dogodila na pozicijama iz skupa β. Sada možemo sumirati prethodni izraz tako što ćemo ga svesti do istog zajedničkog sadržaoca (zajednički sadržalac za međusobno proste polinome u imeniocu je proizvod tih polinoma):

$$\mathbf{S}(x)=\frac{\sum_{i\in\beta}(x^{2w}a_i^{-2w}-1)\prod_{\substack{j\in\beta\\ j\neq i}}(x-a_j)}{\prod_{i\in\beta}x-a_i}=\frac{x^{2w}\sum_{i\in\beta}a_i^{-2w}\prod_{\substack{j\in\beta\\ j\neq i}}(x-a_j)-\sum_{i\in\beta}\prod_{\substack{j\in\beta\\ j\neq i}}(x-a_j)}{\prod_{i\in\beta}x-a_i}.$$

Ova relacija se sada može zapisati kao:

$$\mathbf{S}(x)=\frac{x^{2w}\mathbf{u}(x)}{\mathbf{l}(x)}-\frac{\mathbf{w}(x)}{\mathbf{l}(x)},\ \text{odnosno}\ \mathbf{S}(x)\mathbf{l}(x)=x^{2w}\mathbf{u}(x)-\mathbf{w}(x),$$

gdje su:

$$\mathbf{l}(x)=\prod_{i\in\beta}x-a_i \qquad \mathbf{u}(x)=\sum_{i\in\beta}a_i^{-2w}\prod_{\substack{j\in\beta\\ j\neq i}}(x-a_j) \qquad \mathbf{w}(x)=\sum_{i\in\beta}\prod_{\substack{j\in\beta\\ j\neq i}}(x-a_j).$$

Imamo jednačinu sa tri suštinski nepoznata polinoma, ali od njih nas zapravo interesuje samo jedan. To je polinom $\mathbf{l}(x)$ čije nule su a_i, i to za ono i koje pripada skupu β; odnosno, nule ovog polinoma na jedinstven način označavaju poziciju gdje se greška dogodila. Podsjetimo se da je pretpostavka da su svi a_i različiti. Stoga se ovaj polinom naziva i polinomom mjesta pogreški ili polinomom lokatorom greški. Preostala dva polinoma kod binarnih kodova nisu od interesa, jer ako znamo poziciju pogreške, umijemo i da izvršimo ispravku. Polinom $\mathbf{u}(x)$ (ponekad se naziva polinomom vrednovanja pogreške) je interesantan kod nebinarnih kodova jer nam daje i težinu greške, pa na osnovu polinoma $\mathbf{l}(x)$ i $\mathbf{u}(x)$ možemo da vršimo ispravku pogreški i kod nebinarnih kodova. Opet napominjemo da ćemo se, za sada, zadržati na binarnim kodovima, dok će u Poglavlju VIII biti nešto rečeno i o dekodiranju nebinarnih kodova. Ostaje da vidimo kako iz prethodne jednačine (gdje je poznato samo $\mathbf{S}(x)$ i to da kod može da ispravi do w pogreški) možemo da odredimo $\mathbf{l}(x)$.

VI.6.3 Euklidski algoritam kod BCH kodova

Uočimo da se relacija

$$\mathbf{S}(x)\mathbf{l}(x) = x^{2w}\mathbf{u}(x) - \mathbf{w}(x)$$

može napisati i kao:

$$\mathbf{w}(x) = x^{2w}\mathbf{u}(x) - \mathbf{S}(x)\mathbf{l}(x),$$

te da podsjeća na linearnu kombinaciju kojom možemo zapisati najveći zajednički djelilac dva broja ili polinoma:

$$\gcd = u\ a + v\ b.$$

Do najvećeg zajedničkog djelioca i težinskih koeficijenata dolazili smo provodeći prošireni Euklidski algoritam (pogledati prethodnu sekciju, odnosno Poglavlje I.6.2). Ništa nas ne sprečava da nad polinomima x^{2w} i $\mathbf{S}(x)$ (sindromskim polinomom koji smo naučili da sračunamo) provedemo Euklidski algoritam za polinome sa koeficijentima iz binarnog polja. Međutim, ovaj problem ima već unaprijed poznato i lako uočljivo rješenje, jer ako je polinom $\mathbf{S}(x)$ sa nenultim koeficijentom uz x^0, očigledno je da je najveći zajedniči djelilac x^{2w} i $\mathbf{S}(x)$ jednak 1. Međutim, prošireni Euklidski algoritam nećemo provoditi na posmatrane polinome do kraja, već samo dok stepenom polinoma ostatka r_k ne postane manji od w. Tada u koloni $\mathbf{V}$, u tabeli koju koristimo za provođenje proširenog Euklidskog algoritma, dobijamo:

$$\mathbf{v}_k(x) = \lambda\ \mathbf{l}(x),$$

gdje je λ multiplikativna konstanta različita od 0. Dakle, Euklidski algoritam nam determiniše polinom lokator greške! Pored toga, za datu kolonu važe i sljedeće relacije od značaja za nebinarni kod: $\mathbf{r}_k(x) = \lambda\ \mathbf{w}(x)$ i $\mathbf{u}_k(x) = \lambda\ \mathbf{u}(x)$. Prije nego pokažemo i dokažemo ovu činjenicu, moramo pokazati da su $\mathbf{l}(x)$ i $\mathbf{u}(x)$ međusobno prosti, odnosno da im je najveći zajednički djelilac 1. Podsjetimo se $\mathbf{l}(x)$ i $\mathbf{u}(x)$:

$$\mathbf{l}(x) = \prod_{i\in\beta} x - a_i \qquad\qquad \mathbf{u}(x) = \sum_{i\in\beta} a_i^{-2w} \prod_{\substack{j\in\beta \\ j\neq i}} (x - a_j).$$

Potencijalni djelioci $\mathbf{u}(x)$ su polinomi oblika $(x - a_l)$ za $l \in \beta$. Polinom $\mathbf{u}(x)$ je djeljiv sa $(x - a_l)$ ako je $\mathbf{u}(a_l) = 0$. Međutim ovo nije ispunjeno jer:

$$\mathbf{u}(a_l) = \sum_{i\in\beta} a_i^{-2w} \prod_{\substack{j\in\beta \\ j\neq i}} (a_l - a_j) = a_l^{-2w} \prod_{\substack{j\in\beta \\ j\neq i}} (a_l - a_j)$$

predstavlja proizvod nenultih elemenata koji daje nenultu vrijednost. U Tabeli VI.1 prikazali smo dešavanje kod proširenog Euklidskog algoritma u redu prije

$\mathbf{q}_{j-1}(x)$	$\mathbf{r}_{j-1}(x)$	$\mathbf{u}_{j-1}(x)$	$\mathbf{v}_{j-1}(x)$
$\mathbf{q}_j(x)$	$\mathbf{r}_j(x)=\mathbf{w}^*(x)$	$\mathbf{u}_j(x)=\mathbf{u}^*(x)$	$\mathbf{v}_j(x)=\mathbf{l}^*(x)$

Tabela VI.1. Stanje u tabeli koja se koristi kod proširenog Euklidskog algoritma u redu u kom stepen polinoma ostatka padne ispod w

nego stepen polinoma $\mathbf{r}_k$ padne ispod w. Za svaki red tabele, pa i posljednji može se primijeniti relacija koja povezuje sadržaj kolona **U** i **V** sa polaznim polinomom:

$$\mathbf{w}^*(x)=\mathbf{u}^*(x)x^{2w}+\mathbf{l}^*(x)\mathbf{S}(x).$$

Ako je $\mathbf{l}^*(x)=\lambda\ \mathbf{l}(x)$ dobili smo metodologiju za određivanje polinoma lokatora greške. Prije nego to pokažemo direktnim putem, moramo prvo da pokažemo da je stepen polinoma $\mathbf{l}^*(x)$ manji ili jednak od w, dok su stepeni polinoma $\mathbf{u}^*(x)$ i $\mathbf{w}^*(x)$ manji od w:

$$\deg(\mathbf{l}^*(x))\le w \qquad \deg(\mathbf{u}^*(x))<w \qquad \deg(\mathbf{w}^*(x))<w.$$

Treća relacija koja se odnosi na polinom $\mathbf{w}^*(x)$ predstavlja, zapravo, uslov za zaustavljanje algoritma, tako da ovo nije potrebno dokazivati. Na uzastopne kolone matrice može se primijeniti relacija:

$$\mathbf{l}^*(x)\mathbf{r}_{j-1}(x)-\mathbf{v}_{j-1}(x)\mathbf{w}^*(x)=\pm x^{2w}.$$

Po istoj logici po kojoj je amplituda u koloni **R** opadala kod određivanja najvećeg zajedničkog djelioca brojeva i amplituda elemenata u kolonama **U** i **V** rasla po istoj logici i elementi u kolonama **U** i **V** imaju rastuće stepene, a elementi u koloni **R** imaju opadajuće stepene. Sada je lako uočiti da je stepen prvog člana u izrazu na lijevoj strani veći od izraza na desnoj strani, pa slijedi da je:

$$\deg(\mathbf{l}^*(x)\mathbf{r}_{j-1}(x))=2w.$$

Kako je po uslovima algoritma $\deg(\mathbf{r}_{j-1}(x))\ge w$, to je $\deg(\mathbf{l}^*(x))\le w$. Na sličan se način iz izraza:

$$\mathbf{u}^*(x)\mathbf{r}_{j-1}(x)-\mathbf{u}_{j-1}(x)\mathbf{w}^*(x)=\pm\mathbf{S}(x)$$

može dokazati da je $\deg(\mathbf{u}^*(x))<w$. Sada pođimo od početnog reda tabele i od reda sa polinomima koji su označeni sa *:

$$\mathbf{u}(x)x^{2w}+\mathbf{l}(x)\mathbf{S}(x)=\mathbf{w}(x)$$
$$\mathbf{u}^*(x)x^{2w}+\mathbf{l}^*(x)\mathbf{S}(x)=\mathbf{w}^*(x).$$

Kada pomnožimo prvu jednačinu sa $\mathbf{l}^*(x)$, a drugu sa $\mathbf{l}(x)$ i oduzmemo ove dvije jednačine, dobijamo:

$$(\mathbf{l}^*(x)\mathbf{u}(x)-\mathbf{l}(x)\mathbf{u}^*(x))x^{2w}=\mathbf{l}^*(x)\mathbf{w}(x)-\mathbf{l}(x)\mathbf{w}^*(x).$$

Na lijevoj strani očigledno imamo polinom reda koji je jednak ili veći od $2w$. Međutim, na desnoj strani imamo polinom koji je manji od $2w$, što se može jednostavno pokazati. Da bi prethodna jednakost važila, lijeva i desna strana izraza moraju biti jednake nuli, odnosno:

$$\mathbf{l}^*(x)\mathbf{u}(x)=\mathbf{l}(x)\mathbf{u}^*(x)$$
$$\mathbf{l}^*(x)\mathbf{w}(x)=\mathbf{l}(x)\mathbf{w}^*(x).$$

Veoma smo blizu da dokažemo da se Euklidski algoritam može koristiti za dekodiranje BCH koda. Već smo pokazali da je najveći zajednički djelilac za polinome $\mathbf{l}(x)$ i $\mathbf{u}(x)$ jednak 1. Ovo znači da postoje polinomi $\mathbf{f}(x)$ i $\mathbf{g}(x)$ takvi da važi:

$$\mathbf{f}(x)\mathbf{l}(x)+\mathbf{g}(x)\mathbf{u}(x)=1.$$

Pomnožimo obje strane ove relacije sa $\mathbf{l}^*(x)$ i zamijenimo $\mathbf{l}^*(x)\mathbf{u}(x)$ sa $\mathbf{l}(x)\mathbf{u}^*(x)$:

$$\mathbf{l}^*(x)=(\mathbf{f}(x)\mathbf{l}^*(x)+\mathbf{g}(x)\mathbf{u}^*(x))\mathbf{l}(x)=\mathbf{k}(x)\mathbf{l}(x).$$

Slično se može dobiti da je $\mathbf{w}^*(x)=\mathbf{k}(x)\mathbf{w}(x)$ kao i $\mathbf{u}^*(x)=\mathbf{k}(x)\mathbf{u}(x)$. Kako je $\mathbf{u}_j(x)=\mathbf{u}^*(x)$ i $\mathbf{v}_j(x)=\mathbf{l}^*(x)$, te kako se ± 1 može napisati kao linearna kombinacija $\mathbf{u}_j(x)$ i $\mathbf{v}_j(x)$, slijedi da je najveći zajednički djelilac $\mathbf{u}^*(x)$ i $\mathbf{v}^*(x)=1$. Odavde bi se moglo pokazati da važi, recimo, da je $\mathbf{u}^*(x)=\mathbf{z}(x)\mathbf{u}(x)$, a ovo je moguće samo ako je polinom $\mathbf{k}(x)$ multiplikativna konstanta, odnosno $\mathbf{k}(x)=\lambda$ i $\mathbf{z}(x)=1/\lambda$. Ovim smo dokazali da Euklidski algoritam, primijenjen na polinome do reda koji zadovoljava uvedeni uslov, određuje polinom lokator greške sa tačnošću do muliplikativne konstante (kod binarnih kodova ova konstanta može biti samo ± 1) $\mathbf{l}^*(x)=\lambda\ \mathbf{l}(x)$. Nule ovog polinoma odgovaraju pozicijama na kojima su se dogodile pogreške. Obratite pažnju da smo na početku izvođenja, prilikom određivanja sindromskog polinoma, uzeli negativne stepene pojedinih kolona, pa to ima uticaj na pozicije grešaka u datom polinomu (praktično je riječ o obrnutim pozicijama).

VI.6.4 Algoritam BCH dekodiranja

Dajmo kratak algoritamski opis procesa BCH dekodiranja primjenom Euklidskog algoritma.

1. Sračunati koeficijente sindromskog polinoma: $S_j=\sum_{i=0}^{n-1} r_i a_i^{-j},\ j=1,2,...,2w.$
2. Provesti prošireni Euklidski algoritam za x^{2w} i $\mathbf{S}(x)=S_1+S_2x+...+S_{2w}x^{2w-1}$ i algoritam zaustaviti kada stepen polinoma $\mathbf{r}_j(x)$ padne ispod w. Postaviti $\mathbf{l}(x)=\mathbf{v}_j(x)$ i $\mathbf{u}(x)=\mathbf{u}_j(x)$.

3. Odrediti β skup nula polinoma **l**(x).

4. Naći oblik pogreške $\mathbf{e} = [e_0, e_1, ..., e_{n-1}]$, $e_i = 0$ ako $a^{-i} \notin \beta$ i $e_i = 1$ ako $a^{-i} \in \beta$.

5. Procijenjena kodna riječ je: $\mathbf{c}' = \mathbf{r} - \mathbf{e}$. U slučaju nebinarnog koda, potrebno je na osnovu **u**(x) sračunati težine pogreški da bi se obavilo uspješno dekodiranje. Mi to nećemo raditi.

Važno je napomenuti da Euklidski algoritam nije i jedini koji se može koristiti za BCH dekodiranje; međutim, zadržaćemo se na ovom postupku, bez ulaženja u druge alternative koje su dostupne u literaturi.

VI.6.5 Primjer dekodiranja

Primjer VI.8. Neka je dat kod BCH(15,3) sa prostim polinomom $x^4 + x + 1$. Izvršiti dekodiranje primljene riječi $\mathbf{r} = [111000110011110]$.

Rješenje: Sindrom ima oblik:

$$S_j = 1 + a^j + a^{2j} + a^{6j} + a^{7j} + a^{10j} + a^{11j} + a^{12j} + a^{13j}.$$

Koeficijenti sindromskog polinoma su: $S_1 = a^7$, $S_2 = a^{14}$, $S_3 = a^{11}$, $S_4 = a^{13}$, $S_5 = 1$ i $S_6 = a^7$. Primijenimo Euklidski algoritam na polinome x^6 i $\mathbf{S}(x) = a^7 + a^{14}x + a^{11}x^2 + a^{13}x^3 + x^4 + a^7x^5$. Koraci u algoritmu su dati u Tabeli VI.2. U tabeli smo označili u uglastim zagradama, radi pojednostavljenja, stepene k od a^k, uz odgovarajuće koeficijente u polinomima, s tim da su izostavljeni stepeni (sa koeficijentom 0) označeni sa *. Tako je $a^{11}x^4 + a^9x^3 + a^2x^2 + a^8$ označeno kao [11,9,2,*,8]. Algoritam je zaustavljen kod $i = 3$, jer je $\deg(r_3) < 3$, pa važi: $\mathbf{l}(x) = \mathbf{v}_3(x) = a^7x^3 + a^5x^2 + a^8x + a$. Algoritam za dekodiranje pokazuje da $\mathbf{l}(x) = 0$ ima tri rješenja, i to: $x = a^3, a^9, a^{12}$. Napominjemo da se nule mogu tražiti putem direktne pretrage u programskoj petlji i da složenost procesa ne može biti veća od dužine kodne riječi. Postoje, naravno, načini da se to malo ubrza, posebno kada se problem svede na polinom drugog reda, tipa Vietovih formula. Naime, suma stepena preostalih nula jednaka je stepenu uz koeficijent najnižeg reda. Pozicije pogreški su $(n+1) - l$, gdje su $l \in \beta$ stepeni polinoma lokatora greške. Ovo važi za

I	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[7,0,13,11,14,7]	...
1	[8,1]	[11,9,2,*,8]	[8,1]
2	[4,2,*]	[8,6,9,*]	[11,14]
3	[7,5,8,1]	[7,*,8]	[3,*]

Tabela VI.2. Provođenje Euklidskog algoritma za Primjer VI.7 (izostavljena je **U** kolona pošto kod binarnih kodova nije od značaja)

i_1	i_2	i_3	q_1
i_4	i_5	i_6	q_2
q_3	q_4	q_5	q_6

1	1	0	0
0	1	1	0
1	0	1	0

Tabela VI.3. Pogodan prikaz pravougaonog koda (12,6) i procedura kodiranja zadate poruke

sve stepene $l \neq 0$, dok bi za $l = 0$ bila u pitanju prva pozicija u kodnoj riječi. Stoga zaključujemo da su se pogreške u našem kodu dogodile na pozicijama 13, 7, 4, pa konačno možemo procijeniti da je ispravna kodna riječ:

$$\mathbf{c}' = \mathbf{r} - \mathbf{e} = [111000110011110] + [000100100000100] = [111100010011010].$$

Dekodirana poruka je [1 0 0 1 0] (zapamtite da je primljena poruka „obrnuta“, pa je u istom smislu „obrnuta“ i dekodirana poruka).

VI.7 Zadaci i softverska realizacija

VI.7.1 Riješeni zadaci

6.1. Poruku 110011 treba kodirati sa pravougaonim kodom (12,6). Bitovi parnosti se postavljaju na posljednjim pozicijama u kodu. Odrediti kontrolnu matricu ovog koda. Do greške u rezultujućoj poruci je došlo na poziciji 7. Odrediti sindrom i uporediti s odgovarajućom kolonom kontrolne matrice.

Rješenje: Pravougaoni kod (12,6) koristi ukupno $n = 12$ bita, od čega su $k = 6$ informacioni biti. Pogodan način zapisa dat je u Tabeli VI.3, pri čemu su sa i označeni informacioni biti, a sa q kontrolni biti. U istoj tabeli prikazan je postupak kodiranja zadate poruke. Kodirana poruka je sada:

i_1	i_2	i_3	i_4	i_5	i_6	q_1	q_2	q_3	q_4	q_5	q_6
1	1	0	0	1	1	0	0	1	0	1	0

.

Za kreiranje kontrolne matrice potrebno je znati koje informacione bite kontrolišu pojedini kontrolni biti:

q_1 – kontroliše bite i_1, i_2, i_3 $\quad$ q_2 – kontroliše bite i_4, i_5, i_6
q_3 – kontroliše bite i_1, i_4 $\quad$ q_4 – kontroliše bite i_2, i_5
q_5 – kontroliše bite i_3, i_6 $\quad$ q_6 – kontroliše bite i_1, i_2, i_3, i_4, i_5, i_6.

Stoga je kontrolna matrica sljedećeg oblika:

$$i_1 \;\; i_2 \;\; i_3 \;\; i_4 \;\; i_5 \;\; i_6 \;\; q_1 \;\; q_2 \;\; q_3 \;\; q_4 \;\; q_5 \;\; q_6$$

$$\mathbf{H}=\begin{bmatrix}1&1&1&0&0&0&1&0&0&0&0&0\\0&0&0&1&1&1&0&1&0&0&0&0\\1&0&0&1&0&0&0&0&1&0&0&0\\0&1&0&0&1&0&0&0&0&1&0&0\\0&0&1&0&0&1&0&0&0&0&1&0\\1&1&1&1&1&1&0&0&0&0&0&1\end{bmatrix}.$$

Primljena poruka sa greškom na 7. bitu je:

i_1	i_2	i_3	i_4	i_5	i_6	q_1	q_2	q_3	q_4	q_5	q_6
1	1	0	0	1	1	1	0	1	0	1	0

.

Sindrom **S** nam daje poziciju greške i određuje se kao proizvod primljene poruke **c** i transponovane kontrolne matrice **H**:

$$\mathbf{S}=\mathbf{cH}^{\mathrm{T}}=[1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0]\times$$

$$\begin{bmatrix}1&1&1&0&0&0&1&0&0&0&0&0\\0&0&0&1&1&1&0&1&0&0&0&0\\1&0&0&1&0&0&0&0&1&0&0&0\\0&1&0&0&1&0&0&0&0&1&0&0\\0&0&1&0&0&1&0&0&0&0&1&0\\1&1&1&1&1&1&0&0&0&0&0&1\end{bmatrix}^{\mathrm{T}}=$$

$$=[1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0].$$

Ovim smo dobili vrijednost koja odgovara 7. koloni kontrolne matrice **H**, pa zaključujemo da je greška nastala na bitu q_1. Podsjetimo da se operacije obavljaju sabiranjem po modulu dva (odnosno, ekskluzivno ili operacija).

6.2. (a) Dat je trougaoni kod (15,10). Izvršiti kodiranje poruke 1101101010. Bitove parnosti postaviti na posljednjim pozicijama u kodu. (b) Odrediti kontrolnu matricu za ovaj kod. (c) Do greške u prenosu je došlo na poziciji 6. Izvršiti dekodiranje poruke na osnovu kontrolne matrice. (d) Odrediti generatorsku matricu ovog koda.

1	1	0	1	1
1	0	1	1	
0	1	0		
0	0			
0				

Tabela VI.4. Formiranje trougaonog koda (15,10) iz zadatka 6.2

Rješenje: (a) Kod ćemo formirati tako što ćemo formirati trougaonu tabelu (Tabela VI.4). Kodna riječ je: [110110101011000].

(b) Kontrolna matrica ovog koda je:

$$\mathbf{H}=\begin{bmatrix}1&1&1&1&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&1&1&1&1&0&0&0&0&1&0&0&0\\0&0&1&0&0&0&1&1&1&0&0&0&1&0&0\\0&1&0&0&0&1&0&0&1&1&0&\mathbf{0}&0&1&0\\1&0&0&0&1&0&0&1&0&1&0&0&0&0&1\end{bmatrix}.$$

(c) Dobijena poruka sa greškom na šestom bitu je sada: [110111101011000]. Sindrom je sada:

$$\mathbf{cH}^{\mathrm{T}}=\mathbf{S}=[0\ 1\ 0\ 0\ 1].$$

Sada možemo da dekodiramo poruku kao [110110101011000], a nakon ekstrakcije informacionih bita dobijamo: [1101101010].

(d) Generatorska matrica ovog koda je dimenzija 10×15:

$$\mathbf{G}=\begin{bmatrix}1&0&0&0&0&0&0&0&0&0&1&0&0&0&1\\0&1&0&0&0&0&0&0&0&0&1&0&0&1&0\\0&0&1&0&0&0&0&0&0&0&1&0&1&0&0\\0&0&0&1&0&0&0&0&0&0&1&\mathbf{1}&0&0&0\\0&0&0&0&1&0&0&0&0&0&0&1&0&0&1\\0&0&0&0&0&1&0&0&0&0&0&1&0&1&0\\0&0&0&0&0&0&1&0&0&0&0&1&1&0&0\\0&0&0&0&0&0&0&1&\mathbf{0}&0&0&0&1&0&1\\0&0&0&0&0&0&0&0&1&0&0&0&1&1&0\\0&0&0&0&0&0&0&0&0&1&0&0&0&1&1\end{bmatrix}.$$

6.3. (a) Dat je pravougaoni kod (9,4). Izvršiti kodiranje poruke 1101. Bitove parnosti postaviti na posljednjim pozicijama u kodu. (b) Odrediti kontrolnu matricu za ovaj kod. (c) Do greške u prenosu je došlo na poziciji 6. Izvršiti dekodiranje poruke na osnovu kontrolne matrice. (d) Fomirati generatorsku matricu predmetnog koda.

Rješenje: (a) Po potrebi možete raditi ovaj zadatak korak po korak, ali ćemo ovdje ubrzati rješavanje, a na vama je da izvršite provjeru. Kodna riječ je sljedećeg oblika:

1 1 0 1 0 1 1 0 1.

(b) Kontrolna matrica ovog koda je:

$$\mathbf{H} = \left[\begin{array}{cccc|ccccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

(c) Nakon prenosa kanalom, poruka koja je primljena sa pogreškom je:

$$1 \quad 1 \quad 0 \quad 1 \quad 0 \quad \underline{\mathbf{0}} \quad 1 \quad 0 \quad 1.$$

Dobijeni sindrom je jednak [0 1 0 0 0], što ukazuje da se pogreška dogodila na šestoj poziciji, odnosno da je ispravna kodna riječ:

$$1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1.$$

Nakon ekstrakcije informacionih bita, dobijamo riječ: 1101.

(d) Generatorska matrica za ovaj kod je:

$$\mathbf{G} = \left[\begin{array}{cccc|ccccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{array}\right].$$

6.4. Kod vrši provjeru parnosti nad n bita. Neka je $n = 100$ i vjerovatnoća greške na bitu $p = 0.001$. Greške su i.i.d. Odrediti vjerovatnoću da nema greške u poruci, da je greška detektovana i da greška nije detektovana. Ako je vjerovatnoća greške na bitu $p = 0.01$, koliko je potrebno n da bi se postigla vjerovatnoća nedetektovane pogreške manja od 0.005? Koliko je maksimalno n koje ovo omogućava? Za veoma malo p i veliko n odrediti aproksimativnu vjerovatnoću da nema greške u poruci.

Rješenje: Vjerovatnoća da nema pogreške je jednaka:

$$(1-p)^n = 0.9048.$$

Vjerovatnoća da je greška detektovana jednaka je vjerovatnoći da u kodu postoji neparan broj pogreški, što u konkretnom slučaju znači:

$$\sum_{r=1}^{50} \binom{n}{2r-1} p^{2r-1}(1-p)^{101-2r} \approx 0.0907.$$

Vjerovatnoća da se greška ne detektuje jednaka je vjerovatnoći da postoji paran broj pogreški:

$$\sum_{r=1}^{50}\binom{n}{2r}p^{2r}(1-p)^{100-2r}\approx 0.0045.$$

Za $p=0.01$ najveći broj za koji se postiže vjerovatnoća nedetektovane pogreške koja je manja od 0.05 je $n=10$, za koju ova vjerovatnoća iznosi 0.042:

$$\binom{10}{2}p^2(1-p)^8+\binom{10}{4}p^4(1-p)^6+\binom{10}{6}p^6(1-p)^4+\binom{10}{8}p^8(1-p)^2+p^{10}=0.042.$$

Vjerovatnoća da nema greške u sistemu je:

$$(1-p)^n=1-np+n(n-1)p^2/2+\ldots\approx 1-np.$$

6.5. Koja je vjerovatnoća da Hemingov kod sa k informacionih bita neće moći da ispravi grešku u prenosu? Greške na svim bitima su jednake i međusobno nezavisne. Kolika je vjerovatnoća da Hemingov kod za ispravljanje jedne i detekciju dvije pogreške: (a) nema grešaka u prenosu; (b) ispravi jednu pogrešku; (c) detektuje dvije pogreške; (d) ne može da posluži svojoj svrsi?

Rješenje: Pojednostavimo priču na taj način da kod ima n bita. Vjerovatnoća za slučaj da nema pogreške je $(1-p)^n$, dok je vjerovatnoća da se desi jedna greška i da je ona koriguje $np(1-p)^{n-1}$; konačno, vjerovatnoća detekcije dvije pogreške je $n(n-1)p^2\ (1-p)^{n-2}/2$. U slučaju većeg broja pogreški, kod neće moći služiti svojoj svrsi. Međutim, tu se razlikuje slučaj parnog broja pogreški, kada sistem detektuje da je došlo do greške, dok kod neparnog broja pogreški sistem ispravlja „jednu pogrešku“. Konačno, ako je n paran broj i na svim bitima imamo pogrešku, ta će se poruka shvatiti kao ispravna. Za vježbu, formulišite izraze za navedene vjerovatnoće.

6.6. Informacionu riječ [0 1 0 1 1 0 1 1 1 0 0] kodirajmo putem Hemingovog (15,11) koda. Promijeniti jedan bit i prikazati proceduru za određivanje na kojoj se poziciji dogodila greška. Proceduru ponoviti u dva slučaja, kada su bitovi parnosti na pozicijama 1, 2, 4 i 8 i kada su to posljednja četiri bita u poruci.

Rješenje: Neka je informaciona riječ $\mathbf{i}=$[0 1 0 1 1 0 1 1 1 0 0]. U prvoj varijanti kodna riječ je:

$$\mathrm{c}=[\ \_\_\_\ \ \_\_\_\ 0\ \_\_\_\ 1\ 0\ 1\ \_\_\_\ 1\ 0\ 1\ 1\ \ 1\ 0\ 0].$$

Prvi bit kontroliše neparne pozicije, pa je jednak 1, drugi bit kontroliše pozicije 2, 3, 6, 7, 10, 11, 14, 15 i bit je jednak 1, treći bit parnosti kontroliše pozicije 4, 5, 6, 7, 12, 13, 14 i 15, pa je ovaj bit 0, dok posljednji bit parnosti kontroliše krajnjih

8 bita i iznosi 0. Dakle, kodna riječ je: $\mathbf{c} = [1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0]$. Pretpostavimo da se greška dogodila na šestoj poziciji u kodu, pa da je primljena riječ:

$$\mathbf{r} = [1\ 1\ 0\ 0\ 1\ \underline{\mathbf{1}}\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0].$$

Kontrolna matrica ovog koda je:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 \end{bmatrix}.$$

Sindrom je jednak:

$$\mathbf{S} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = [0\ 1\ 1\ 0]^{\mathrm{T}}.$$

Jasno se pokazuje da je greška na šestom bitu jer je sindrom jednak šestoj koloni. Ispravljena riječ je: $\mathbf{c} = [1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0]$, a dekodirana riječ je:

$$\mathbf{i} = [0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0].$$

Posmatrajmo sada varijantu da su simboli parnosti na posljednjim pozicijama u kodu. Kontrolna matrica ovog koda ima permutovane kolone u odnosu na prethodnu, sa mogućim oblikom:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{0} & 0 & 0 & 1 \end{bmatrix}.$$

Ako je informaciona riječ ista kao u prethodnom slučaju, $\mathbf{i} = [0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0]$, kodna riječ je: $\mathbf{c} = [0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0]$. Pretpostavimo pogrešku na šestom bitu: $\mathbf{r} = [0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0]$. Sindrom u ovom slučaju je:

$$\mathbf{S} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = [0\ 1\ 0\ 1]^{\mathrm{T}}.$$

Ponovo zaključujemo da se pogreška dogodila na šestoj poziciji u kodu koji možemo da ispravimo i pravilno dekodiramo kao $[0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0]$.

6.7. Posmatrajte Hemingov kod sa 4 informaciona bita koji može da ispravi jednu i detektuje dvije greške. Kreirati kontrolnu matricu koda. Uzeti četiri proizvoljna bita i kreirati odgovarajući Hemingov kod. Zatim promijeniti jedan, pa dva bita, respektivno, i prikazati način dekodiranja ovog koda.

Rješenje: Pošto drugačije nije specificirano, uzećemo da su biti parnosti na pravilnim binarnim pozicijama. Kontrolna matrica koda u ovom slučaju je:

$$\mathbf{H}=\begin{bmatrix}1&0&1&0&1&0&1&0\\0&1&1&0&0&1&1&0\\0&0&0&1&1&1&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}.$$

Uzmimo da je informaciona riječ $\mathbf{i}=[1\ 0\ 1\ 1]$. Tada je kodna riječ $\mathbf{c}=[0\ 0\ 1\ 0\ 0\ 1\ 1\ 1]$. Promijenimo prvi bit $\mathbf{r}=[1\ 0\ 1\ 0\ 0\ 1\ 1\ 1]$. Pomnožimo ovu kodnu riječ sa kontrolnom matricom, čime dobijamo sindrom:

$$\mathbf{S}=\mathbf{r}\mathbf{H}^{\mathrm{T}}=[1\ 0\ 0\ 1]^{\mathrm{T}}.$$

Kako je broj jedinica u kodnoj riječi neparan, znamo da postoji jedna pogreška. Pogreška se dogodila na prvom bitu pošto je dobijeni sindrom jednak prvoj koloni kontrolne matrice. Sada pretpostavimo da se pogreška dogodila na drugoj i petoj poziciji. Primljena riječ je: $\mathbf{r}=[0\ 1\ 1\ 0\ 1\ 1\ 1\ 1]$, pa je sindrom jednak:

$$\mathbf{S}=\mathbf{c}\mathbf{H}^{\mathrm{T}}=[1\ 1\ 1\ 0]^{\mathrm{T}}.$$

Prvo imamo paran broj jedinica, a to znači da, iako je sindrom nenulti, onda postoji paran broj pogreški. Ujedno vidimo da sindrom nije jednak nijednoj od kolona kontrolne matrice što dalje ukazuje na činjenicu da postoje pogreške, ali da ne možemo da ih ispravimo.

6.8. Poruka je kodirana trougaonim kodom (15,10) i na prijemu glasi: 101101000101100. Biti parnosti su dati na posljednjim mjestima u kodnoj riječi. Izvršiti dekodiranje primljene poruke.

Rješenje: Dekodiranje će biti obavljeno putem matrice. Kontrolna matrica koda je:

$$\mathbf{H}=\begin{bmatrix}1&1&1&1&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&1&1&1&1&0&0&0&0&1&0&0&0\\0&0&1&0&0&0&1&1&1&0&0&0&1&0&0\\0&1&0&0&0&1&0&0&1&1&0&\mathbf{0}&0&1&0\\1&0&0&0&1&0&0&1&0&1&0&0&0&0&1\end{bmatrix}.$$

Sindrom je jednak:

$$\mathbf{S}=\mathbf{r}\mathbf{H}^{\mathrm{T}}=[1\ 0\,1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,1\,0\ 0]\mathbf{H}^{\mathrm{T}}=[1\,1\,0\,0\,0].$$

Sindrom korespondira sa četvrtim bitom riječi, pa nakon ispravljanja dobijamo informacionu riječ:

$$101\underline{\mathbf{0}}010001.$$

i_1	i_2	i_3	i_4	c_1
i_5	i_6	i_7	i_8	c_2
c_3	c_4	c_5	c_6	c_7

Tabela VI.5. Raspored informacionih i kontrolnih bita u pravougaonom kodu (15,8)

6.9. Dat je pravougaoni kod (15,8) gdje su informacioni biti postavljeni u dvije vrste od po 4 elementa. Biti parnosti su postavljeni na posljednja mjesta u kodnoj riječi. Prikazati kontrolnu i generatorsku matricu ovog koda. Primljena je poruka 011111011010111. Dekodirati je!

Rješenje: Pogledajmo Tabelu VI.5. Podrazumijevajući da se biti parnosti nalaze na posljednjim pozicijama u kodnoj riječi, kontrolna matrica dimenzija 7×15 ovog koda ima oblik:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Generatorska matrica koda je dimenzija 8×15 i ima oblik:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{0} & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{0} & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Za sekvencu $\mathbf{r} = [0111\ 1101\ 1010111]$ sindrom $\mathbf{S} = \mathbf{r}\mathbf{H}^{\mathrm{T}}$ iznosi:

$$\mathbf{S} = [0\ 1\ 0\ 0\ 0\ 1\ 1].$$

Kako je sindrom jednak osmoj koloni kontrolne matrice, zaključujemo da se na toj poziciji dogodila pogreška, pa da je poslata riječ $\mathbf{c} = [0111\ 1100\ 1010111]$, te da je informacija $\mathbf{i} = [0111\ 1100]$. Na ovaj način izvršili smo dekodiranje primljene poruke.

6.10. Simboli iz alfabeta $X = \{A, B, C, D\}$ pojavljuju se sa vjerovatnoćama, redom: $\{0.4, 0.3, 0.2, 0.1\}$. Hafmenovim kodom kodirati poruku **DACA**, a zatim je propustiti kroz Hemingov kod (7,4), sa bitovima parnosti na posljednjim pozicijama. U slučaju da nedostaju biti za formiranje pune riječi Hemingovog koda, dopuniti poruku nulama.

Rješenje: Mogući Hafmenov kod za predmetni alfabet je:

$$A \quad 0 \qquad B \quad 10 \qquad C \quad 110 \qquad D \quad 111.$$

Riječ **DACA** se stoga kodira kao: 11101100. Ovu riječ dijelimo na dva bloka od četiri bita 1110 i 1100. Generatorska matrica Hemingovog koda može da ima oblik:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Dva bloka od po četiri bita se stoga mogu kodirati kao:

$$\mathbf{c}_1 = \mathbf{i}_1\mathbf{G} = \begin{bmatrix} 1\,1\,1\,0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = [1\,1\,1\,\mathbf{0}\,0\,0\,0]$$

$$\mathbf{c}_2 = \mathbf{i}_2\mathbf{G} = \begin{bmatrix} 1\,1\,0\,0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = [1\,1\,0\,\mathbf{0}\,0\,1\,1].$$

Dakle, dobijena poruka je na kraju 11100001100011.

6.11. Dokazati da ako je binarni polinom (polinom sa koeficijentima $\{0, 1\}$) prost

$$\mathbf{c}(x) = \sum_{k=0}^{n-1} a_k x^k,$$

tada je prost i polinom:

$$\mathbf{c}(x) = \sum_{k=0}^{n-1} a_k x^{n-k-1}.$$

Rješenje: Dokaz se može provesti intuitivno. Prvo je potrebno dokazati da ako imamo proizvod dva polinoma $\mathbf{a}(x)\mathbf{b}(x)$ koji je jednak nekom polinomu:

$$\mathbf{a}(x)\mathbf{b}(x) = \sum_{p=0}^{P} \alpha_p x^p,$$

tada proizvod polinoma s obrnutim koeficijentima (označimo ih sa $\overline{\mathbf{a}}(x)$ i $\overline{\mathbf{b}}(x)$) daje rezultujući polinom s obrnutim koeficijentima:

$$\overline{\mathbf{a}}(x)\overline{\mathbf{b}}(x) = \sum_{p=0}^{P} \alpha_{P-p} x^p.$$

Sada možemo zaključiti da ako $\sum_{p=0}^{P} \alpha_p x^p$ nije prost, odnosno ako ima djelioce, onda ni polinom $\sum_{p=0}^{P} \alpha_{P-p} x^p$ nije prost. Stoga ako je $\sum_{p=0}^{P} \alpha_p x^p$ prost, onda je prost i $\sum_{p=0}^{P} \alpha_{P-p} x^p$.

6.12. Dokazati da ako polinom ima nenulte koeficijente samo uz parne stepene ne može biti prost.

Napomena. Zadaci 6.11 i 6.12 pomažu da se redukuje pretraga za prostim polinomima.

Rješenje: Opisani polinom se može prikazati kao:

$$\mathbf{c}(x) = \sum_{p=0}^{P} a_p x^{2p}.$$

Posmatrajmo sada polinom:

$$\mathbf{q}(x) = \sum_{p=0}^{P} a_p x^p$$

i izvršimo njegovo kvadriranje:

$$\mathbf{q}^2(x) = \sum_{p_1=0}^{P} \sum_{p_2=0}^{P} a_{p_1} a_{p_2} x^{p_1+p_2} = \sum_{p=0}^{P} a_p^2 x^{2p} + 2\sum_{p_1=0}^{P} \sum_{\substack{p_2=0 \\ p_1 \neq p_2}}^{P} a_{p_1} a_{p_2} x^{p_1+p_2}.$$

Druga suma u ovom izrazu, u posmatranoj aritmetici, jednaka je nuli, dok je $a_P^2 = a_P$ ($0^2 = 0$ i $1^2 = 1$), pa važi:

$$\mathbf{q}^2(x) = \sum_{p=0}^{P} a_p x^{2p}.$$

Dakle, ovim smo dokazali da se binarni polinom sa nenultim koeficijentima samo uz članove uz parne stepene može prikazati kao kvadrat nekog drugog polinoma, to jest da zasigurno nije ni nesvodiv ni prost.

6.13. Prikazati kodere Hemingovog koda koji je dobijen na osnovu binarnog prostog polinoma x^3+x^2+1. Za proizvoljni informacioni polinom provjeriti rad sistema ako se greška dogodi na poziciji uz stepen x^2.

Rješenje: Samostalno kreirajte obje varijante kodera i sa množenjem polinoma i sa povratnom spregom. Konsultujte realizacije koje su date u knjizi. Posmatrajmo sada slučaj da je informaciona riječ 1011, odnosno da ako uzmemo da je prvi bit onaj najmanje važnosti imamo polinom $1+x^2+x^3$, pa kada to pomnožimo sa generatorskim polinomom koji je ponovo x^3+x^2+1, dobijamo: x^6+x^4+1, odnosno 1 0 1 0 0 0 1. Greška na x^2 simbolu daje primljenu poruku 1 0 1 0 1 0 1, odnosno $x^6+x^4+x^2+1$. Kako je ostatak pri dijeljenju ovog polinoma sa generatorskim polinomom baš x^2, dolazimo do zaključka da se pogreška dogodila na navedenom bitu.

6.14. Dokazati da je polinom $x^4+x^3+x^2+x+1$ nesvodiv, ali da nije prost i da nije pogodan za formiranje korespondentnog Hemingovog koda.

Rješenje: Premda smo postupak već ranije obavili, podsjetimo se osnovnih koraka u njemu. Provjera djeljivosti sa polinomima prvog reda x i $x+1$ obavlja se trivijalno. Postoji koeficijent uz x^0 i polinom ima neparan broj članova pa ne može biti djeljiv sa polinomima prvog reda. Ostaje samo provjera da li je djeljiv sa jedinim prostim polinomom drugog reda x^2+x+1. Količnik pri dijeljenju je očigledno x^2 s ostatkom $x+1$, pa stoga zaključujemo da je predmetni polinom nesvodiv (nema smisla dijeliti ga sa polinomima trećeg reda jer bi u tom slučaju rezultat bio polinom prvog reda, a djeljivost sa polinomima prvog reda je već provjerena). Formirajmo, na osnovu ovog nesvodivog polinoma, bazu stepena nule prostog polinoma na način kako smo to radili s drugim prostim polinomima $a^0=[0\ 0\ 0\ 1]^T$, $a^1=[0\ 0\ 1\ 0]^T$, $a^2=[0\ 1\ 0\ 0]^T$, $a^3=[1\ 0\ 0\ 0]^T$. Nula stepena a^4 se dobija na osnovu relacije $\mathbf{p}(a)=a^4+a^3+a^2+a+1=0$, odnosno $a^4=a^3+a^2+a+1=[1\ 1\ 1\ 1]^T$. Naredni stepen $a^5=a^4a=a^4+a^3+a^2+a=\ a^3+a^2+a+1+a^3+\ a^2+a=1$. Dakle, ovo polje se ne zatvara, tako da je n najniži red za koji se stepen nule prostog polinoma svodi na a^0, odnosno ne odgovara dužini kodne riječi koja je projektovana u našem slučaju a^{15}. Stoga je navedeni polinom neupotrebljiv za formiranje Hemingovog koda (15,11), ali, kao što smo vidjeli, ima primjenu kod kodova koji su u stanju da isprave više pogreški.

6.15. Posmatrajte Hemingov kod za korekciju dvije greške, nastao korištenjem polinoma x^4+x+1. Uzmite proizvoljnu binarnu poruku i kodirajte je ovim kodom. Koliko bita mora imati ta binarna poruka? Generišite jednu grešku i ispitajte što ste dobili. Zatim generišite dvije greške i pogledajte kako možete detektovati dobijenu pogrešku.

Rješenje: Kao što smo vidjeli, prosti polinom $x^4 + x + 1$ daje Hemingov kod (15,11), a ako želimo da izvršimo njegovo proširenje tako da bude u stanju da detektuje dvije pogreške, dobijamo kod (16,11). Kontrolna matrica ovog koda je:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & \mathbf{0} & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Binarna poruka kodirana ovim kodom očigledno mora da ima 16 bita (11 informacionih i 5 kontrolnih). Kada se dogodi jedna pogreška, imamo da je sindrom jednak koloni matrice; međutim, u slučaju dvije pogreške, npr. na šestom i jedanaestom bitu, dobijamo sindrom koji je jednak:

$$\mathbf{S} = [1\ 0\ 1\ 0\ 1]^{\mathrm{T}} + [0\ 0\ 1\ 1\ 1]^{\mathrm{T}} = [1\ 0\ 0\ 1\ 0]^{\mathrm{T}}.$$

Kao što se može uočiti, ne postoji navedena kolona koja odgovara sindromu. Dakle, u ovom slučaju i na osnovu kontrolne matrice i određivanja sindroma možemo zaključiti da postoji više od jedne pogreške i da je ne možemo dekodirati. Stoga zaključujemo da je greška detektovana, ali ne i korigovana.

6.16. Neka je broj bita sa kojima želimo kodirati poruku $n = 7$. Koliko informacija možemo poslati kanalom ako želimo da nam kod ispravi dvije greške? Da li je moguća konstrukcija koda koji daje toliko informacija? Ako je moguća, prikažite taj kod, a ako nije, objasnite zbog čega, po vašem mišljenju, nije.

Rješenje: Ukupan broj riječi koje se mogu konstruisati sa $n = 7$ bita je $2^7 = 128$. Oko svake legitimne kodne riječi sada možemo da formiramo sferu sa riječima koje se od nje razlikuju na jednom, odnosno dva bita i nijedan takav skup ne treba da ima presjek. Svaka od sfera ima, dakle, $1 + n + n(n-1)/2$ članova, što u našem slučaju iznosi $1 + 7 + 21 = 29$. Stoga, možemo zaključiti da na osnovu pakovanja sfera možemo da konstruišemo 128/29, odnosno, 4 kodne riječi. Postavlja se pitanje: da li je ovo stvarno moguće? Pretpostavimo da imamo binarnu kodnu riječ *abcdefg* i da imamo kodnu riječ koja se od nje razlikuje na 5 pozicija (tolika Hemingova distanca je potrebna da bismo dobili mogućnost prepoznavanja dvije pogreške) $ab\bar{c}\bar{d}\bar{e}fg$, gdje smo sa $\bar{x}$ označili komplement bita (bez gubitka opštosti uzeto je da su prva dva bita ista). Postavlja se pitanje: da li postoji kodna riječ sa sedam bita koja se od obje prethodne legitimne kodne riječi razlikuje za 5 mjesta? Najveća moguća razlika na prva dva bita je dva i ona se postiže ako je početak treće kodne riječi $\bar{a}\bar{b}$. Dakle, u preostalih 5 bita potrebno je postići razliku od 3 bita u odnosu na obje riječi, što je očigledno nemoguće; ako su tri nekomplemen-

tirane, onda će razlika sa prvom kodnom riječju biti samo dva, dok ako su tri komplementirane, onda će razlika sa drugom kodnom riječi biti samo dva. Stoga možemo, na osnovu ovog prostog eksperimenta, da zaključimo da je moguće imati najviše dvije kodne riječi za koje važi navedeno pravilo. Stoga je uputno ili skratiti kodnu riječ na pet bita, jer nam omogućava istu fleksibilnost vezanu za broj riječi s ispravkom dvije pogreške, ili se prosto opredijeliti da vršimo ispravku 3 pogreške sa kodnom riječju od sedam bita. Zbog čega je ovo tako? Odgovorićemo na ovo pitanje sa manje matematičkog formalizma, ali na prost način. Posmatrajmo kvadrat stranice a, njegova površina je a^2. Uzmimo kvadrat stranice $a\sqrt{2}/2$, čija je površina $a^2/2$. Poređenjem površina mogli bismo da zaključimo da možemo smjestiti dva manja u jedan veći kvadrat, ali svi znamo da to nije moguće. Ista je situacija i kod kodova gdje zbog geometrije prostora nije uvijek moguće idealno (a ponekad ni približno idealno) spakovati sfere. Očigledno da nam Hemingova distanca i prateća granica ne daju dovoljno informacija o mogućnostima konstrukcije kodova. Stoga smo u Poglavlju VIII dali nekoliko detalja o ovom složenom pitanju granica koje se mogu postići.

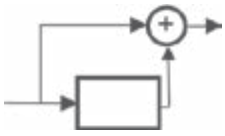
6.17. Kod se generiše tako što se k informacionih bita preslikava na prvih k bita kodne riječi, nakon čega se dodaju redundantni biti tako da kodna riječ bude djeljiva bez ostatka sa polinom $\mathbf{p}(x)=x+1$. Kakav je kod u pitanju? Koliki je broj bita u kodnoj riječi i kodni odnos? Prikazati hardversku strukturu koja realizuje predmetni kod. Da li je predmetni kod cikličan (dokazati)?

Rješenje: Ako informacionih bita ima k (polinom reda $k-1$), množenjem sa posmatranim polinomom dobićemo polinom k-tog reda, odnosno $k+1$ bit kodne riječi. Dakle, dati kod može da predstavlja sistem sa provjerom parnosti, dimenzija $(k+1,k)$.

Stoga zaključujemo da su performanse ovog koda ekvivalentne kodu sa provjerom parnosti. Hardverska struktura ovog sistema je prikazana na Slici VI.7. Cikličnost ovog koda možemo provjeriti dijeljenjem x^2+1 sa polinomom $\mathbf{p}(x)$, čime se lako pokazuje da je rezultat jednak $x+1$ bez ostatka. Posmatrajmo sada informacioni polinom:

$$\mathbf{i}(x)=\sum_{i=0}^{k-1} i_k x^i .$$

Množenjem sa generatorskim polinomom dobijamo:



Slika VI.7. Koder za kod iz zadatka 6.17

$$\mathbf{c}(x) = \mathbf{i}(x)\mathbf{p}(x) = i_{k-1}x^k + \sum_{i=0}^{k-2}[i_{i+1} + i_i]x^{i+1} + i_0.$$

Uočavamo da su samo dva bita direktno predstavljena u kodnoj riječi (prvi i posljednji), dok su svi ostali prikazani kao zbirovi drugih simbola. Međutim, nije teško zaključiti da je prvih k bita zapravo Grejov kod informacione sekvence, dok se posljednji može zamijeniti sumom svih informacionih simbola, pa je predmetni kod ekvivalentan provjeri parnosti. Dekodiranje se može obaviti kao Grejovo dekodiranje od početka (ili kraja) kodne riječi i poređenjem posljednjeg dekodiranog bita sa posljednjim bitom riječi.

6.18. Odrediti dimenzije koda, kontrolne i generatorske matrice i kodni odnos koji ima pet informacionih simbola i vrši četiri provjere parnosti. Ponoviti postupak sa kodom minimalne dužine koji ima pet informacionih simbola i koji je u stanju da ispravi četiri pogreške.

Rješenje: U prvom slučaju kod ima $n = 5 + 4$ bita, pa je riječ o blok kodu (9,5). Kodni odnos je $R = 5/9$. Generatorska matrica je dimenzija 5×9, dok je kontrolna matrica dimenzija 4×9.

Kod drugog koda vidjeli smo da je kod koji je u stanju da ispravi tri pogreške sa pet informacionih bita blok kod (15,5). Da bi ispravio još jednu pogrešku, mora da bude primijenjen još jedan polinom četvrtog reda tako da su minimalne dimenzije blok koda koji ima 5 informacionih bita i ispravlja četiri greške (19,5), odakle slijedi da je kodni odnos $R = 5/19$, dimenzija generatorske matrice 5×19 i dimenzija kontrolne matrice 14×19.

6.19. Hemingov kod (15,11) formiran je putem prostog polinoma $x^4 + x^3 + 1$. Koliki je kodni odnos ovog koda? U slučaju da je izvršeno kodiranje putem množenja polinoma, te da je primljena poruka:

$$[1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1],$$

izvršiti dekodiranje ove poruke putem polinoma. Na kraju kontrolne riječi dodat je bit parnosti za čitavu riječ, čime je dobijen kod (16,11). Koliki je kodni odnos ovog koda i koja je minimalna Hemingova distanca za ovaj kod? Prikazati kontrolnu matricu ovog proširenog koda.

Rješenje: Kodni odnos koda (15,11) je $R = 11/15$. Kontrolna matrica ovog koda je:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & \mathbf{0} & 0 & 0 & 1 \end{bmatrix}.$$

Generatorska matrica je:

$$\mathbf{G}=\begin{bmatrix}1&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\0&1&0&0&0&0&0&0&0&0&0&0&1&1&0\\0&0&1&0&0&0&0&0&0&0&0&0&0&1&1\\0&0&0&1&0&0&0&0&0&0&0&1&1&0&1\\0&0&0&0&1&0&0&0&0&0&0&1&0&1&0\\0&0&0&0&0&1&0&0&0&0&0&0&1&0&1\\0&0&0&0&0&0&1&0&0&0&0&1&1&1&0\\0&0&0&0&0&0&0&1&0&0&0&0&1&1&1\\0&0&0&0&0&0&0&0&1&0&0&1&1&1&1\\0&0&0&0&0&0&0&0&0&1&0&1&0&1&1\\0&0&0&0&0&0&0&0&0&0&1&1&0&0&1\end{bmatrix}.$$

Primljena riječ se može prikazati putem polinoma:

$$\mathbf{r}(x)=x^{14}+x^{12}+x^{10}+x^9+x^8+x^6+x^5+1.$$

Dijeljenjem ovog polinoma sa generatorskim polinomom, dobijamo količnik:

$$\begin{aligned}x^{14}+x^{12}+x^{10}+x^9+x^8+x^6+x^5+1 &: x^4+x^3+1 = x^{10}+x^9+x^4+x^3+x\\ x^{13}+x^{12}+\quad x^9+x^8+x^6+x^5+1&\\ x^8+x^6+x^5+1&\\ x^7+x^6+x^5+x^4+1&\\ x^5+x^4+x^3+1&\\ x^3+x+1.&\end{aligned}$$

Dakle, ostatak pri dijeljenju je x^3+x+1. Sada treba da provjerimo kojem stepenu nule prostog polinoma odgovara ovaj ostatak. Za trenutak ćemo pretpostaviti da ne raspolažemo sa kontrolnom matricom iz koje se odmah vidi na kojoj je poziciji nastala pogreška:

$$a^{14}=a^{-1}=a^3+a^2.$$

Pretpostavimo da je naš ostatak jednak $a^d=a^3+a+1$, pa ga pomnožimo sa a^{-1}:

$$a^d a^{-1}=(a^3+a^2)(a^3+a+1)=a^6+a^4+a^3+a^5+a^3+a^2=a^6+a^4+a^5+a^2.$$

Uočimo da važi: $a^4=a^3+1$, $a^5=a^4+a=a^3+a+1$, $a^6=a^4+a^2+1=a^3+a^2+a$

$$a^d a^{-1}=a^3+1=a^4.$$

Dakle, jednostavno zaključujemo da se pogreška dogodila na poziciji koja odgovara stepenu $d-1=4$, odnosno $d=5$. Tako da je ispravna kodna poruka:

$$x^{14}+x^{12}+x^{10}+x^9+x^8+x^6+1 : x^4+x^3+1 = x^{10}+x^9+x^4+x^3+1$$
$$x^{13}+x^{12}+\quad x^9+x^8+x^6+1$$
$$x^8+x^6+1$$
$$x^7+x^6+x^4+1$$
$$x^4+x^3+1$$

prikazana u obliku vektora: [11000011001].

6.20. Kodiranje je izvršeno putem Hemingovog ternarnog koda (4,2) sa generatorskim polinomom x^2+2x+2. Dekodirati poruku [2 1 0 1] i putem kontrolne matrice i putem polinoma.

Rješenje: Usvajajući da su dva stepena nule prostog polinoma:

$$a^0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad a^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

uočimo da važi:

$$a^2 = -2a - 2 = a + 1.$$

Dakle, $a^2 = a+1$, odnosno $a^3 = a \cdot a^2 = a(a+1) = a^2 + a = 2\mathit{0} + 1$:

$$a^2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad a^1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Kontrolna matrica koda je:

$$\mathbf{H} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

Sindrom je jednak:

$$\mathbf{S} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = [2\ 1\ 0\ 1] \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 4+1 & 2+1+1 \end{bmatrix} = [5\ \ 4] = [2\ 1].$$

Greška se dogodila na prvom bitu i težine je 1, pa je ispravna poruka [1 1 0 1]. Ako podijelimo ovaj polinom sa generatorskim polinomom, dobijamo:

$$x^3+x^2+\quad 1 : x^2+2x+2 = x+2$$
$$x^3+2x^2+2x$$
$$2x^2+x+1.$$

Odavde slijedi da je poslata poruka [1 2]. Direktnom primjenom polinoma dobijamo (nakon dijeljenja):

$$\begin{array}{l} 2x^3+x^2+\quad 1 : x^2+2x+2 = 2x \\ x^3+x^2+x \\ \qquad\quad 2x+1. \end{array}$$

Ostatak je $2x+1$, a direktno uočavamo da je ostatak jednak x^3, te da se greška dogodila na prvom bitu i da je težine 1. Stoga se dekodiranje dalje vrši kao što je već prethodno urađeno.

6.21. Kreirati hardversku strukturu koja kodira ternarni Golajev kod (11,6). Dokazati da je ovaj kod cikličan.

Rješenje: Generatorski polinom Golajevog ternarnog koda je: $\mathbf{f}(x) = x^5 + x^4 + 2x^3 + x^2 + 2$. Hardverska realizacija je prikazana na Slici VI.8.

Kod je cikličan ako je polinom $x^n - 1$ (u našem slučaju $x^{11} - 1 = x^{11} + 2$) djeljiv sa generatorskim polinomom:

$$\begin{array}{l} (x^{11}+2) : (x^5+x^4+2x^3+x^2+2) = x^6+2x^5+2x^4+2x^3+x^2+1 \\ \underline{x^{11}+x^{10}+2x^9+x^8+2x^6} \\ 2x^{10}+x^9+2x^8+x^6+2 \\ \underline{2x^{10}+2x^9+x^8+2x^7+x^5} \\ 2x^9+x^8+x^7+x^6+2x^5+2 \\ \underline{2x^9+2x^8+x^7+2x^6+x^4} \\ 2x^8+2x^6+2x^5+2x^4+2 \\ \underline{2x^8+2x^7+x^6+2x^5+x^3} \\ x^7+x^6+2x^4+2x^3+2 \\ \underline{x^7+x^6+2x^5+x^4+2x^2} \\ x^5+x^4+2x^3+x^2+2. \end{array}$$

Pošto smo dobili djeljivost bez ostatka možemo zaključiti da je u pitanju ciklični kod.

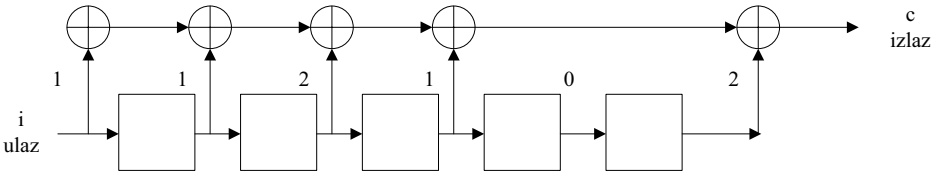
6.22. Formirati kontrolnu matricu koda sa $r = 5$ simbola. U kodnoj riječi ima $n - k = 2$ provjere parnosti.

Rješenje: Kontrolna matrica koja predstavlja jedno moguće rješenje postavljenog problema je:

$$\mathbf{H} = \begin{bmatrix} 4 & 3 & 2 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Prvo treba provjeriti da li ovakav kod može da postoji. Kao što smo vidjeli, između dužine kodne riječi, broja informacionih simbola i veličine alfabeta važi sljedeća relacija:

$$n = \frac{r^{n-k} - 1}{r - 1}.$$



Slika VI.8. Model hardverske realizacije Golajevog (11,6) koda

Uvrštavanjem gorenavedenih parametara dobijamo da je $n=6$, pa je zatim kontrolna matrica dimenzija $(n-k)\times n = 2\times 6$ baš kao što je u ovom slučaju naša **H** matrica. Predmetna matrica predstavlja kontrolnu matricu koda, a generatorska se može dobiti kao:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 4 \\ 0 & 1 & 0 & 0 & 2 & 4 \\ 0 & 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 0 & 1 & 4 & 4 \end{bmatrix}.$$

Postavlja se pitanje: kakve su karakteristike ovog koda u pogledu mogućnosti za ispravljanje pogreški? To pitanje, uz primjer realizacije, prepuštamo čitaocima.

6.23. Data je kontrolna matrica ternarnog Golajevog (11,6) koda:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 1 & 2 & 0 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Odrediti generatorsku matricu koda. Kodirati riječ [0, 1, 1, 2, 0, 2], unijeti pogrešku težine 2 na poziciju 8 i dekodirati.

Rješenje: Generatorska matrica koda je:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 & 2 \\ 0 & 1 & 0 & 0 & 0 & 0 & 2 & 2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 2 & 1 & 2 \\ 0 & \mathbf{0} & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 2 & 2 \end{bmatrix}.$$

Za datu informaciju dobija se kodna riječ:

$$\mathbf{c} = \mathbf{iG} = [0\,1\,1\,2\,0\,2]\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 2 & 1 & 1 \end{bmatrix} = [0\,1\,1\,2\,0\,2\,0\,0\,2\,0\,1].$$

Unesimo sada grešku na poziciji 8, težine 2:

$$\mathbf{r} = \mathbf{c} + [0\,0\,0\,0\,0\,0\,0\,2\,0\,0\,0] =$$
$$= [0\,1\,1\,2\,0\,2\,0\,0\,2\,0\,1] + [0\,0\,0\,0\,0\,0\,0\,2\,0\,0\,0] = [0\,1\,1\,2\,0\,2\,0\,2\,2\,0\,1].$$

Odredimo sindrom množenjem ovog vektora sa kontrolnom matricom:

$$\mathbf{S} = \mathbf{rH}^{\mathrm{T}} = [0\,2\,0\,0\,0]^{\mathrm{T}}.$$

Ovim smo pokazali da se greška dogodila na osmoj poziciji, sa težinom dva, jer dobijeni sindrom korespondira dvostrukoj osmoj koloni kontrolne matrice. Problem dekodiranja je ovim riješen jer se greška nalazi na poziciji kontrolnog simbola, pa je prvih šest simbola informaciona poruka.

6.24. Provjeriti da li je polinom $x^5 + x^2 + 1$ prost. Zatim provjeriti da li može poslužiti za stvaranje koda dužine 31. Ako može, iskoristiti ga za kreiranje BCH koda odgovarajuće dužine kodne riječi koji može da ispravi 2, odnosno 3 pogreške.

Napomena. Može da se dogodi da kod ovog polinoma ne dobijete da su polinomi $\mathbf{p}_3(x)$ i $\mathbf{p}_5(x)$ sa svim jediničnim koeficijentima.

Rješenje: Provjera da li je predmetni polinom prost može da se obavi jednostavnim dijeljenjem ovog polinoma sa x, $x+1$ i x^2+x+1, odnosno sa prostim (nesvodivim) polinomima nižeg reda od 3. Očigledna je činjenica da polinom nije djeljiv sa x (posjeduje slobodni član) i da nije djeljiv sa $x+1$ jer bi u tom slučaju za $x=1$ polinom x^5+x+1 trebao da bude jednak nuli, što nije slučaj. Jedino je dijeljenjem potrebno provjeriti da li je posmatrani polinom djeljiv sa x^2+x+1 (polinomom drugog reda):

$$\begin{array}{l} x^5 + x^2 + 1 : x^2 + x + 1 = x^3 + x^2 \\ \underline{x^5 + x^4 + x^3} \\ \quad x^4 + x^3 + x^2 + 1 \\ \quad \underline{x^4 + x^3 + x^2} \\ \qquad\qquad 1. \end{array}$$

Ne treba dalje provjeravati sa polinomima višeg reda, npr. trećeg, jer bismo dobili polinom drugog reda, a provjeru djeljivosti sa polinomima drugog reda smo već

obavili. Da bi predmetni polinom bio odgovarajući generatorski polinom potrebno je da ne postoji stepen manji od $r = 2^5 - 1 = 31$, takav da važi da je $x^r + 1$ djeljivo sa posmatranim polinomom. Predmetna osobina je zadovoljena, ali vam je dokaz prepušten zbog obima. Kontrolna matrica koda (31,26) koji se može generisati putem predmetnog polinoma je:

$$\mathbf{H} = \begin{bmatrix} 1&0&1&0&1&1&1&0&1&1&0&0&0&1&1&1&1&1&0&0&1&1&0&1&0&0&1&0&0&0&0 \\ 0&1&0&1&0&1&1&1&0&1&1&0&0&0&1&1&1&1&1&0&0&1&1&0&1&\mathbf{0}&0&1&0&0&0 \\ 0&0&1&0&1&0&1&1&1&0&1&1&0&0&0&1&1&1&1&1&0&0&1&1&0&1&0&0&1&0&0 \\ 1&0&1&1&1&0&1&1&0&0&0&1&1&1&1&1&0&0&1&1&\mathbf{0}&1&0&0&1&0&0&0&0&1&0 \\ 0&1&0&1&1&1&0&1&1&0&0&0&1&1&1&1&1&0&0&1&1&0&1&0&0&1&0&0&0&0&1 \end{bmatrix}.$$

Odredimo sada redove polinoma $\mathbf{p}_3(x)$ i $\mathbf{p}_5(x)$. Konjugati za $\mathbf{p}_3(x)$ su $\{3, 6, 12, 24, 48 = 48\%31 = 17\}$. Sljedeći konjugat je 34, a on je jednak $34\%31 = 3$, pa možemo zaključiti da je 5 različitih konjugata broja 3, odnosno da je $\mathbf{p}_3(x)$ polinom petog reda. Konjugati petice su $\{5, 10, 20, 40 = 40\%31 = 9, 18\}$, dok bi sljedeći konjugat bio $36 = 36\%31 = 5$, odnosno ponovljen prvi element skupa, pa zaključujemo da je i ovaj polinom petog stepena. Sada je potrebno da odredimo koeficijente ovih polinoma. Neka su:

$$\mathbf{p}_3(x) = p_{35}x^5 + p_{34}x^4 + p_{33}x^3 + p_{32}x^2 + p_{31}x + p_{30}$$
$$\mathbf{p}_5(x) = p_{55}x^5 + p_{54}x^4 + p_{53}x^3 + p_{52}x^2 + p_{51}x + p_{50}.$$

Uvrstimo $x = a^3$ u prvi izraz i $x = a^5$ u drugi. Trebalo bi da dobijemo nula vektore:

$$\mathbf{p}_3(a^3) = p_{35}[11111] + p_{34}[01110] + p_{33}[11010] +$$
$$+ p_{32}[01010] + p_{31}[01000] + p_{30}[00001] = [00000]$$
$$\mathbf{p}_5(a^5) = p_{55}[11001] + p_{54}[01100] + p_{53}[11111] +$$
$$+ p_{52}[10001] + p_{51}[00101] + p_{50}[00001] = [00000].$$

Odavde slijedi da je $p_{35} + p_{33} = 0$, a po istoj logici, koju smo ranije objašnjavali, slijedi da su $p_{35} = 1$ i $p_{33} = 1$. Kako je $p_{35} + p_{30} = 0$, slijedi da je $p_{30} = 1$. Dalje, imamo da je $p_{35} + p_{34} = 1$ (za srednji bit), pa je odatle $p_{34} = 1$. Za četvrti bit od početka slijedi $p_{35} + p_{34} + p_{33} + p_{32} = 0$, a to dalje znači da je $p_{32} = 1$. Konačno, za prvi bit od početka slijedi $p_{35} + p_{34} + p_{33} + p_{32} + p_{31} = 0$, a to onda znači da je $p_{31} = 0$ (ne jedan kao u dosadašnjim primjerima). Time dobijamo da je $\mathbf{p}_3(x)$ jedan od prostih polinoma 5. reda:

$$\mathbf{p}_3(x) = x^5 + x^4 + x^3 + x^2 + 1.$$

Što se tiče polinoma $\mathbf{p}_5(x)$, da bismo pojednostavili potragu, možemo poći od činjenice da je on zasigurno polinom petog reda, što implicira da je $p_{55} = 1$. Uvidom u ostale jednačine slijedi da je $p_{53} = 0$ (za četvrti bit od početka), a to dalje znači da je $p_{54} = 1$, $p_{52} = 1$, $p_{51} = 1$ i $p_{50} = 1$. Na ovaj način ponovo dobijamo jedan od prostih polinoma petog reda:

$$\mathbf{p}_5(x) = x^5 + x^4 + x^2 + x + 1.$$

Generatorski polinom koda BCH(31,3) je:

$$\mathbf{g}(x) = \mathbf{p}(x)\mathbf{p}_3(x)\mathbf{p}_5(x) = (x^5 + x^2 + 1)(x^5 + x^4 + x^3 + x^2 + 1)(x^5 + x^4 + x^2 + x + 1) =$$
$$= x^{15} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1.$$

Za dalji rad vam prepuštamo određivanje odgovarajućeg hardvera za kodiranje ovog koda i provjeru rada sistema i dekodiranje za slučaj od jedne do tri pogreške.

Ako nam je potrebno da sa $n = 31$ bit formiramo BCH kod koji ispravlja četiri pogreške, treba da posmatramo konjugate a^7: (7, 14, 28, 56%31 = 25, 50%31 = 19), dok je naredni 38%31 = 7. Dakle, ponovo imamo 5 različitih konjugata, odnosno polinom petog reda. Dakle, kod koji je u pitanju je (31, 11) kod:

$$\mathbf{p}_7(a^7) = p_{75}[10000] + p_{74}[10110] + p_{73}[11000] + p_{72}[11101] + p_{71}[10100] +$$
$$+ p_{70}[00001] = [00000].$$

Lako možemo poći od pretpostavke da je $p_{75} = 1$ (polinom petog stepena). Odavde slijedi da je $p_{74} + p_{73} + p_{72} + p_{71} = 1$. Pošto je $p_{73} + p_{72} = 0$, prvi uslov svodimo na $p_{74} + p_{71} = 1$. Za treće bite važi da je $p_{74} + p_{72} + p_{71} = 0$, a odavde lako zaključujemo da je $p_{72} = 1$, pa slijedi da je $p_{73} = 1$. Provjerom za četvrte bite imamo $p_{74} = 0$, pa je $p_{71} = 1$. Konačno, provjerom petih bita slijedi da je $p_{70} = 1$. Dakle, polinom $\mathbf{p}_7(x)$ je:

$$\mathbf{p}_7(x) = x^5 + x^3 + x^2 + 1.$$

Generatorski polinom je sada:

$$\mathbf{g}'(x) = \mathbf{g}(x)\mathbf{p}_7(x) = x^{20} + x^{18} + x^{17} + x^{16} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^3 + x + 1.$$

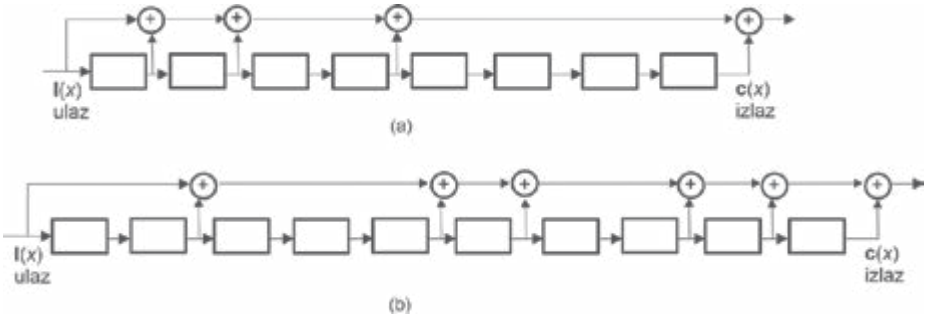
6.25. Odrediti makar jedan polinom pogodan za generisanje BCH koda dužine 255 bita, kao i jedan generatorski polinom za BCH(255,2) kod.

Rješenje: Bez želje da detaljnije ulazimo u ovu problematiku, dajemo osnovni prosti polinom osmog stepena koji se može koristiti za kreiranje BCH kodova sa 255 bita dugom kodnom riječju, uz napomenu da ako koristimo samo ovaj polinom, dobijamo odgovarajući Hemingov kod (255,247):

$$\mathbf{p}(x) = x^8 + x^4 + x^3 + x^2 + 1.$$

Na osnovu predmetnog polinoma, procedurom koja sa porastom reda polinoma i sa dužom kodnom riječju postaje prilično komplikovana za ručnu obradu, dobijamo polinom:

$$\mathbf{g}(x) = x^{16} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x + 1.$$



Slika VI.9. BCH koderi (15,2) i (15,3) dobijeni na osnovu prostog polinoma $x^4 + x^3 + 1$

Dvjesta pedeset i pet bita, koliko je kodna riječ ovog koda, veoma je pogodno jer je broj informacionih bita 239, a to predstavlja (gotovo) 30 riječi ASCII koda po 8 bita, dok se kodiranje vrši putem (približno) 32 kodne riječi po 8 bita.

6.26. Odrediti generatorski polinoma za BCH(15,3), koristivši prosti polinom $x^4 + x^3 + 1$. Prikazati realizaciju pomoću pomjeračkog registra.

Rješenje: Procedura za određivanje predmetnog generatorskog polinoma je ista kao i u slučaju prostog polinoma $x^4 + x + 1$. Ponovo se dobija da su $\mathbf{p}_3(x) = x^4 + x^3 + x^2 + x + 1$ i $\mathbf{p}_5(x) = x^2 + x + 1$ (provjerite). Generatorski polinomi za BCH(15,2) i BCH(15,3) su:

$$\mathbf{g}_1(x) = (x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1) = x^8 + x^4 + x^2 + x + 1$$

$$\mathbf{g}_2(x) = (x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1.$$

Koderi BCH kodova s generatorskim polinoma $\mathbf{g}_1(x)$ i $\mathbf{g}_2(x)$ dati su na Slici VI.9.

6.27. Data je poruka 10111. Kodirati je BCH kodom BCH(15,3) iz prethodnog zadatka. Generisati jednu pogrešku i izvršiti dekodiranje. Generisati dvije pogreške i izvršiti dekodiranje i generisati tri pogreške i izvršiti dekodiranje.

Rješenje: Kodni polinom u ovom slučaju je:

$$\mathbf{c}(x) = \mathbf{i}(x)\mathbf{g}_2(x) = (x^4 + x^2 + x + 1)(x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1) =$$
$$= x^{14} + x^{13} + x^9 + x^6 + x^5 + x^3 + x + 1.$$

Ovu kodnu riječ možemo zapisati kao [110001001101011]. Uzmimo sada da imamo tri slučaja. U prvom, jednu grešku i to na poziciji 5:

[110011001101011].

U drugom slučaju, dvije greške, na pozicijama 10 i 14:

[1100010010010 01].

Konačno, u trećem slučaju imamo tri pogreške, na pozicijama 3, 8 i 12:

[111001011100011].

Koeficijenti sindromskog polinoma u prvom slučaju su:

$$S_j = 1 + a^j + a^{4j} + a^{5j} + a^{8j} + a^{9j} + a^{11j} + a^{13j} + a^{14j}.$$

Pogledajmo sada koeficijente:

$$S_1 = 1 + a + a^4 + a^5 + a^8 + a^9 + a^{11} + a^{13} + a^{14} = a^4$$
$$S_2 = 1 + a^2 + a^8 + a^{10} + a + a^3 + a^7 + a^{11} + a^{13} = a^8$$
$$S_3 = 1 + a^3 + a^{12} + 1 + a^9 + a^{12} + a^3 + a^9 + a^{12} = a^{12}$$
$$S_4 = 1 + a^4 + a^1 + a^5 + a^2 + a^6 + a^9 + a^7 + a^{11} = a^1$$
$$S_5 = 1 + a^5 + a^5 + a^{10} + a^{10} + 1 + a^{10} + a^5 + a^{10} = a^5$$
$$S_6 = 1 + a^6 + a^9 + 1 + a^3 + a^6 + a^6 + a^3 + a^9 = a^9.$$

Tabela VI.6 prikazuje realizaciju Euklidskog algoritma za predmetni slučaj sa jednom pogreškom. Polinom lokator greške je $\mathbf{l}(x) = a^6 x + a^2$, pa je nula ovog polinoma $x = a^2/a^6 = a^{-4} = a^{11}$. Zaključujemo da je pozicija pogreške peta, što odgovara našem eksperimentu.

U drugom slučaju (primljene riječi sa dvije pogreške) koeficijenti sindromskog polinoma su:

$$S_j = 1 + a^j + a^{5j} + a^{8j} + a^{11j} + a^{14j}.$$

Konkretne vrijednosti koeficijenata sindromskog polinoma su, redom:

$$S_1 = a^{12},\ S_2 = a^9,\ S_3 = a^{13},\ S_4 = a^3,\ S_5 = a^{10} \text{ i } S_6 = a^{11}.$$

Tabela VI.7 prikazuje realizaciju Euklidskog algoritma. Polinom lokator greške je $\mathbf{l}(x) = x^2 + a^5 x + a^8$. Njegove nule su $x = a^2$ i $x = a^6$ što ukazuje da su pogreške bile na pozicijama 14 i 10.

U slučaju sa tri pogreške, odgovara primljena poruka:

[111001011100011].

Sindrom je dakle:

$$S_j = 1 + a^j + a^{2j} + a^{5j} + a^{7j} + a^{8j} + a^{9j} + a^{13j} + a^{14j}.$$

Koeficijenti sindromskog polinoma su:

$$S_1 = a^8,\ S_2 = a,\ S_3 = a^3,\ S_4 = a^2,\ S_5 = a^5,\ S_6 = a^6.$$

i	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[4,8,12,1,5,9]	...
1	[6,2]	[6]	[6,2]

Tabela VI.6. BCH(15,3) kod sa prostim polinomom $x^4 + x^3 + 1$ u slučaju jedne pogreške

I	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[11,10,3,13,9,12]	...
1	[4,3]	[0, 5, 2, 0, 0]	[4,3]
2	[0, 5, 8]	[5]	[11,3]

Tabela VI.7. BCH(15,3) kod sa prostim polinomom $x^4 + x^3 + 1$ u slučaju dvije pogreške

i	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[6,5,2,3,1,8]	...
1	[9,8]	[5, 4, 7, 0, 1]	[9,8]
2	[10, 9, 0]	[10, 10, 13, 8]	[1,*]
3	[5, 5, 8, 0]	[5, *, 8]	[10, 6]

Tabela VI.8. BCH(15,3) kod sa prostim polinomom $x^4 + x^3 + 1$ u slučaju tri pogreške

Realizacija Euklidskog algoritma je prikazana Tabelom VI.8. Polinom lokator greške je $\mathbf{l}(x) = a^5x^3 + a^5x^2 + a^8x + 1$. Nule ovog polinoma su $x = a^4$, $x = a^8$ i $x = a^{13}$. Odavde slijedi da su pogreške na pozicijama 12, 8 i 3, odnosno da smo pravilno identifikovali pozicije pogreške.

6.28. BCH(15,3) je konstruisan na osnovu prostog polinoma $x^4 + x^3 + 1$. Primljena riječ je:

1 0 1 0 0 0 0 0 0 0 0 0 1 1 1.

Izvršiti njeno dekodiranje putem Euklidskog algoritma.

Rješenje: Da bismo baratali ovim sindromima i ostalim elementima koda, potrebno je prethodno da konstruišemo kontrolnu matricu predmetnog Hemingovog koda (15,11), konstruisanog datim prostim polinomom:

$$\mathbf{H}=\begin{bmatrix}1&0&0&1&1&0&1&0&1&1&1&1&0&0&0\\1&1&0&1&0&1&1&1&1&0&0&0&1&0&0\\0&1&1&0&1&0&1&1&1&1&0&0&0&1&0\\0&0&1&1&0&1&0&1&1&1&1&\mathbf{0}&0&0&1\end{bmatrix}.$$

Koeficijenti sindromskog polinoma su:

$$S_j=1+a^j+a^{2j}+a^{12j}+a^{14j},\ \ j\in[1,6]$$

$$S_1=1+a+a^2+a^{12}+a^{14}=a^{14}$$

$$S_2=1+a^2+a^4+a^9+a^{13}=a^{13}$$

$$S_3=1+a^3+a^6+a^6+a^{12}=a^7$$

$$S_4=1+a^4+a^8+a^3+a^{11}=a^{11}$$

$$S_5=1+a^5+a^{10}+a^0+a^{10}=a^5$$

$$S_6=1+a^6+a^{12}+a^{12}+a^9=a^{14}.$$

Tabela VI.9 prikazuje korake u Euklidskom algoritmu. Polinom lokator pogreške je:

$$\mathbf{l}(x)=a^2x^3+a^{14}x+1.$$

Nule predmetnog polinoma su a^7, a^{10}, a^{11}, pa zaključujemo da su se pogreške dogodile na pozicijama $n+1-7=9$, $n+1-10=6$, $n+1-11=5$, pa je kodna riječ:

1 0 1 0 **1** **1** 0 0 **1** 0 0 0 1 1 1.

Dekodiranjem (na bilo koji pogodan način) dobijamo da je informaciona riječ:

1 0 0 0 1.

i	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[14,5,11,7,13,14]	...
1	[1,7]	[13, *, 10, 6]	[1,7]
2	[2, *, 14, 0]	[2, *, 14]	[1, 7, *]

Tabela VI.9. Euklidski algoritam za BCH kod iz zadatka 6.28

VI.7.2 *Softverska realizacija*

A. Ilustrujmo rad sa kontrolnom matricom na slučaju Hemingovog (7,4) koda.

```
clear
n=7; k=4;
P=[1 0 1;1 0 1;1 1 0;1 1 1];
I=(rand(1,k)>0.5);  %slučajno odabrani informacioni vektor
H=[P',eye(n−k)];
G=[eye(k),P];
C=rem(I*G,2);  %kodna riječ
[m,p]=max(ceil(n*rand));
%slučajno generisana pozicija pogreške
E=zeros(1,n); E(p)=1;  %vektor pogreške
R=xor(C,E);  %poruka sa jednom pogreškom
S=rem(R*H',2);  %sindrom
pz=1; ind=0;  %potraga za pozicijom pogreške
while ((pz<=n) & (ind==0))
   if(all(S'−H(:,pz)==0))
%nađena kolona koja odgovara sindromu
      ind=1;
   end
   pz=pz+1;
end
pz=pz−1;
R1=R;
R1(pz)=xor(R(pz),1);
ID=R1(1:k);  %dekodirana poruka
disp('informacioni biti')
disp(I)
disp('kodirana poruka')
disp(C)
disp('poruka sa greskom')
disp(R)
disp('sindrom')
disp(S)
disp('ispravljena pogreška')
disp(R1)
disp('dekodirana poruka')
disp(ID)
```

Na početku smo formirali matricu P, a ona nam je poslužila da na osnovu nje kreiramo kontrolnu matricu H i generatorsku matricu G. Informacione bite smo odabrali slučajno (funkcija rand), a zatim smo odredili kodnu riječ množenjem, po modulu 2, informacionog vektora sa generatorskom matricom. Naredne dvije linije služe nam da bismo odredili vektor pogreške (generisali smo samo jednu pogrešku), nakon čega funkcijom xor (ekskluzivno ili po elementima vektora argumenata) dobijamo primljenu kodnu riječ. Na odgovarajući način određujemo sindrom i poredimo ga sa kontrolnom matricom, red po red. Ispravljamo poruku na onoj poziciji u kodnoj riječi gdje smo detektovali da se pogreška dogodila i ispisujemo sve dobijene rezultate. Jedna naša realizacija ovog koda vodila je ka sljedećim rezultatima, s tim da će svaka realizacija, zbog toga što smo slučajno generisali poziciju pogreške i informacione bite, biti različita.

```
informacioni biti
    1     0     1     0
kodirana poruka
    1     0     1     0     0     1     1
poruka sa greskom
    0     0     1     0     0     1     1
sindrom
    1     0     1
ispravljena pogreska
    1     0     1     0     0     1     1
dekodirana poruka
    1     0     1     0
```

Osnovni problem kod ovog pristupa je što matrica P, koja je najvažniji element u ovom kodu, može da bude neprihvatljivo velika. Međutim, kod Hemingovog koda činjenica je da se matrica H sastoji od svih mogućih kombinacija bitova 0 i 1, isključujući kombinaciju sa svim nulama. Napisati program koji je u stanju da rukuje ovim problemom sa što je moguće manje direktnog unošenja matrice P.

B. Implementirajmo kodiranje putem pomjeračkog registra sa povratnom spregom kodne riječi [1 0 0 1] (kao u Poglavlju VI.3.1).

```
clear
info_rijec=[1 0 0 1];
registar=info_rijec;
%% nakon punjenja registra bitima informacione rijeci
for k=1:3
   registar=rem(registar+registar(1)*[1 0 1 1],2);
%%povratna sprega i xor operacija
```

```
    registar=[registar(2:4) 0];
end
kodna_rijec=[info_rijec,registar(2:4)]
1    0    1    0    0    1    1
```

Unesimo grešku na poziciji 4, kodna_rijec(4)=1. Dijeljenje polinoma u MATLAB-u možemo izvršiti naredbom deconv:

```
[q,r]=deconv(kodna_rijec,[1 0 1 1]);
```

Potrebno je vratiti rezultate u domen binarnih brojeva:

```
q=rem(q,2);
r=rem(r,2);
r=r(end−2:end);
```

Posljednja naredba je potrebna kako bi se izbrisao nepotrebni dio polinoma ostatka sa nulama.

```
k=0;
while sum(abs(r−[0 0 1]))~=0
k=k+1;
rc=conv(r,a6);
r=rem(rc(3:5)+rc(2)*[0 1 1]+ rc(1)*[1 1 0],2);
end
```

Nakon k=3 ponavljanja dobili smo da je $a^r = a^0$. To ukazuje da je pozicija pogreške kodna_rijec(end−k)=xor(kodna_rijec(end−k),1). Za vježbu napisati program koji će u opštem slučaju vršiti dekodiranje za različite generatorske polinome. Detaljno protumačiti navedeni program. Napominjemo da je a6=[1 1 0], što odgovara $a^6 = a^{-1}$.

C. Kreirati kontrolnu matricu za kod sa $n = 15$ bita koji je u stanju da ispravi dvije pogreške iz Poglavlja VI.5:

```
H=zeros(4,15);
H(:,15)=[0 0 0 1]';
for r=14:−1:1,H(:,r)=rem([H(2:4,r+1);0]+H(1,r+1)*[0 0 1 1]',2);end
H
  1  1  1  1  0  1  0  1  1  0  0  1  0  0  0
  0  1  1  1  1  0  1  0  1  1  0  0  1  0  0
  0  0  1  1  1  1  0  1  0  1  1  0  0  1  0
  1  1  1  0  1  0  1  1  0  0  1  0  0  0  1
H1=zeros(4,15);
for r=15:−1:1
```

```
H1(:,r)=H(:,15−rem(3*(15−r),15));
end
H=[H;H1];
H =
 1  1  1  1  0  1  0  1  1  0  0  1  0  0  0
 0  1  1  1  1  0  1  0  1  1  0  0  1  0  0
 0  0  1  1  1  1  0  1  0  1  1  0  0  1  0
 1  1  1  0  1  0  1  1  0  0  1  0  0  0  1
 1  1  1  1  0  1  1  1  1  0  1  1  1  1  0
 1  0  1  0  0  1  0  1  0  0  1  0  1  0  0
 1  1  0  0  0  1  1  0  0  0  1  1  0  0  0
 1  0  0  0  1  1  0  0  0  1  1  0  0  0  1
```

Za vježbu napisati program koji je u stanju da dekodira pojavljivanje jedne, odnosno dvije greške kod ove kontrolne matrice.

D. Dobra je prilika da se upoznamo s osnovnim naredbama za rad sa Hemingovim kodom u MATLAB-u. Naime, u MATLAB-u je kreiran komunikacioni alat (engl. *toolbox*) koji omogućava rad s nizom popularnih blok kodova na pojednostavljen način. Ovdje ćemo se upoznati s funkcijama koje se odnose na Hemingov kod. Funkcija hammgen formira kontrolu i generatorsku matricu koda. Argument ove funkcije je broj mjesta parnosti:

```
[kont,gener]=hammgen(3);
gener =
    1    1    0    1    0    0    0
    0    1    1    0    1    0    0
    1    1    1    0    0    1    0
    1    0    1    0    0    0    1
```

Generatorski polinom može se dobiti funkcijom cycpoly čiji su argumenti dužina koda i broj mjesta parnosti:

```
genpoly = cyclpoly(7,3);
```

Funkcija cyclgen sa dva argumenta – dužinom kodne riječi i generatorskim polinomom daje nam kontrolnu i generatorsku matricu koda:

```
[kont,gener]= cyclgen(7,genpoly);
```

Funkcija gen2par konvertuje generatorsku u kontrolnu matricu koda. Napominjemo da je suprotno od MATLAB-ove logike u radu sa polinomima – ovdje generatorski polinomi imaju najniži red na prvom mjestu u vektoru. Ostalo smo već objasnili ranije. Ako posjedujemo primljenu riječ, sindrom se može odrediti prostim množenjem riječi sa kontrolnom matricom po modulu 2:

```
sindrom=rem(c*kont',2);
```

E. Uradimo zadatak 6.28 u MATLAB-u. Kreirajmo prvo matricu P i na osnovu nje kontrolnu H i generatorsku matricu G:

```
P=[1 1 1 2 2 0; 1 1 2 1 0 2; 1 2 1 0 1 2;1 2 0 1 2 1;1 0 2 2 1 1];
H=[P,eye(5)];
G=[eye(6),3−P'];
```

Na osnovu informacionog vektora, kreirajmo kodnu riječ i izvršimo njenu izmjenu da bismo simulirali primljenu riječ:

```
i=[0 1 1 2 0 2];
c=rem(i*G,3)
c =
   0    1    1    2    0    2    0    0    2    0    1
r=c;
r(8)=rem(r(8)+2,3);
```

Sračunajmo sindrom:

```
S=rem(r*H',3)
S =
   0    2    0    0    0
```

Jednostavnom provjerom uočavamo da je sindrom jednak dvostrukoj osmoj koloni, čime smo u prilici da na odgovarajući način ispravimo pogrešku.

F. Za ilustrativne potrebe realizovan je MATLAB program koji prati osnovne korake u kodiranju i dekodiranju BCH(15,3) koda (s oba osnovna prosta polinoma). Realizacija nije optimalna, ali je zgodna za praćenje svih koraka u algoritmu i kao takva je dobra za razumijevanje procesa kodiranja i dekodiranja.

Osnovni program main_bch.m

```
clear
global MM
c=input('Unesi kodni vektor sa 5 bitova ');
disp('Kodna rijec koja se kodira');
disp(c)
pause(1)
tip=input('Ako je gen. polinom x^4+x+1 unijeti 1 a za x^4+x^3+1 unijeti 2 ');
MM=parametri_bch(2,tip);
x=bch(c,tip);
disp('Kodirana kodna rijec');
```

```
disp(x)
pause(1)
izmj=input('Unesi pozicije na kojima se mijenja kodna rijec ');
r=izmjeni(x,izmj);
disp('Izmjenjena kodna rijec: ')
disp(r)
pause(1)
sindrom=sindr_pol(r);
disp('Sindrom ')
help_prikaz(sindrom)
disp('Koraci u euklidovom algoritmu')
ts=euklidovalgoritam(sindrom);
npoz=nule_polin(ts);
disp('Nule polinoma t su: ')
disp(npoz)
pause(1)
npoz=(npoz>0).*(16−npoz)+(npoz== 0);
disp('Sto znaci da su zapravo se greske dogodile: ')
disp(npoz)
pause(1)
xr=izmjeni(r,npoz);
disp('Popravljena kodna poruka: ')
disp(xr)
pause(1)
[ce,ost]=dij_bin_pol(xr(15:−1:1),parametri_bch(1,tip));
ce=[zeros(1,5−length(ce)),ce];
ce=ce(5:−1:1);
if(isempty(ost))
   disp(,Dekodiranje uspjelo ,)
   disp(ce)
   pause(1)
   disp('Uporedi sa polaznim vektorom ')
   disp(c)
   pause(1)
else
  disp('Dekodiranje nije uspjelo')
end
```

Nakon unosa informacionog polinoma, funkcija MM=parametri_bch(2,tip); suštinski formira kontrolnu matricu (ovdje to radimo radi lakšeg obavljanja operacija sabiranja i množenja). Iz navedenih razloga MM je deklarisana kao globalna

promjenljiva. Funkcija bch ima namjenu da kreira kodni polinom. Ova funkcija ima jednostavnu realizaciju:

```
function y=bch(c,tip)
y=rem(conv(c(5:−1:1),parametri_bch(1,tip)),2);
y=y(15:−1:1);
```

Uočimo da smo ovdje koristili funkciju parametri_bch(1,tip) i vidimo da ova funkcija, s prvim argumentom jednakim 1, daje generatorski polinom, dok sa 2 daje generatorsku matricu. Promjenljiva tip odnosi se na to koji je osnovni prosti polinom izabran (x^4+x+1 ili x^4+x^3+1). Nakon toga korisnik zadaje pozicije gdje su se greške dogodile, pa se funkcijom izmijeni vrši modifikacija kodnog polinoma (dogodile su se pogreške u kanalu). Realizacija ove proste funkcije je:

```
function z=izmjeni(y,poz)
e=zeros(1,length(y));
e(1,poz)=1;
z=xor(y,e);
```

Naredbom sindrom=sindr_pol(r); vrši se generisanje sindromskog polinoma:

```
function s=sindr_pol(r)
for k=1:6
   s(2,k)=0;
   for l=1:15
         if(r(l)==1)
            if(s(2,k)==0)
               s(1,k)=rem(k*(l−1),15);
               s(2,k)=1;
            else
               zz=zbirstep(s(1,k),rem(k*(l−1),15));
               if(isempty(zz))
                  s(2,k)=0;
               else
                  s(1,k)=zz;
               end
            end
         end
   end
end
```

Razmotrite sami realizaciju ove funkcije. Unutar ove funkcije koristi se funkcija zbirstep, koja sabira dvije kolone generatorske matrice osnovnog polinoma (onog koji ispravlja jednu pogrešku) i vraća stepen koji odgovara rezultatu:

```
function z=zbirstep(z1,z2);
global MM
x=xor(MM(z1+1,:),MM(z2+1,:));
for k=1:15
   if(sum(abs(x−MM(k,:)))==0)
      z=k−1;
      return
   end
end
z=[];
```

Najsloženiji dio programa je funkcija ts=euklidovalgoritam(sindrom); koja služi za provođenje Euklidskog algoritma:

```
function ts=euklidovalgoritam(sind)
tp=[0;0];
rp=[0 0 0 0 0 0 0;0 0 0 0 0 0 1];
ts=[0;1];
rs=sind;
n=size(rp,2);
while(n>3)
   [kol,ost]=dijeli_polin_f(rp,rs);
   disp('Kolicnik ')
   help_prikaz(kol)
   disp('Ostatak ')
   help_prikaz(ost)
   tr=saberi_polin_f(tp,mnozi_polin_f(kol,ts));
   rp=rs;
   rs=ost;
   tp=ts;
   ts=tr;
   disp('Tekuci vektor t ')
   help_prikaz(ts)
  disp('Tekuci vektor r ')
  help_prikaz(rs)
  n=size(rs,2);
end
```

Ključna funkcija kod Euklidskog algoritma je ona koja vrši dijeljenje polinoma sa koeficijentima koji su stepeni nule prostog polinoma:

```
[kol,ost]=dijeli_polin_f(rp,rs);
```

Realizacija ove funkcije data je kao:

```
function [kol,ost]=dijeli_polin_f(imen,djel)
M=max(find(imen(2,:)==1));
imen=imen(:,1:M);
N=max(find(djel(2,:)==1));
djel=djel(:,1:N);
kol=zeros(2,M−N+1);
k=M−N+1;
while M>=N
   kol(1,M−N+1)=rem(imen(1,M)−djel(1,N)+15,15);
   kol(2,M−N+1)=1;
   imen=saberi_polin_f(imen,mnozi_polin_f(kol(:,1:M−N+1),djel));
   M=max(find(imen(2,:)==1));
   imen=imen(:,1:M);
end
ost=imen;
```

Kod Euklidskog algoritma prikazani su međurezultati putem funkcije help_prikaz jer je ideja da čitalac može da kontroliše svoje rezultate. Funkcije saberi_polin_f i mnozi_polin_f koriste se za sabiranje i množenje polinoma sa koeficijentima koji su stepeni nula prostog polinoma. Realizacija funkcije saberi_polin_f:

```
function q3=saberi_polin_f(q1,q2)
M=size(q1,2); N=size(q2,2);
q3=zeros(2,max(M,N));
for m=1:min(M,N)
   if(q1(2,m)==1 & q2(2,m)==1)
      aq=zbirstep(q1(1,m),q2(1,m));
      if(isempty(aq))
         q3(2,m)=0;
      else
         q3(2,m)=1;
         q3(1,m)=aq;
      end
   elseif(q1(2,m)==1)
      q3(1,m)=q1(1,m);
      q3(2,m)=1;
```

```
    elseif(q2(2,m)==1)
        q3(1,m)=q2(1,m);
        q3(2,m)=1;
    end
end
if(M>N)
    q3(:,min(M,N)+1:M)=q1(:,min(M,N)+1:M);
else
    q3(:,min(M,N)+1:N)=q2(:,min(M,N)+1:N);
end
```

Realizacija funkcije mnozi_polin_f:

```
function q3=mnozi_polin_f(q1,q2)
M=size(q1,2); N=size(q2,2);
q3=zeros(2,M+N−1);
for m=1:M
    for n=1:N
        if(q1(2,m)==1 & q2(2,n)==1 & q3(2,m+n−1)==0)
            q3(1,m+n−1)=rem(q1(1,m)+q2(1,n),15);
            q3(2,m+n−1)=1;
        elseif(q1(2,m)==1 & q2(2,n)==1)
            aq=zbirstep(q3(1,m+n−1),rem(q1(1,m)+q2(1,n),15));
            if(isempty(aq))
                q3(2,m+n−1)=0;
            else
                q3(1,m+n−1)=aq; q3(2,m+n−1)=1;
            end
        end
    end
end
```

Funkcija npoz=nule_polin(ts); daje nule polinoma lokatora pogreške. Realizacija ove funkcije je:

```
function npoz=nule_polin(ts)
pp=0;
for k=0:14
    ind=0;
    for l=1:size(ts,2)
        if(ind==0 & ts(2,l)==1)
            tek=rem(ts(1,l)+(l−1)*k,15);
            ind=1;
```

```
        elseif(ts(2,l)==1)
            tek=zbirstep(tek,rem(ts(1,l)+(l−1)*k,15));
            if(isempty(tek))
                ind=0;
            end
        end
    end
    if(ind==0)
        npoz(pp+1)=k;
        pp=pp+1;
    end
end
    if(pp==0),npoz=[];end
```

Prava pozicija pogreški dobija se kao:

```
npoz=(npoz>0).*(16−npoz)+(npoz==0);
```

Kao što znamo, nule koje dobijamo za a^k za $k>0$ odgovaraju pozicijama $16-k$, dok stepenu a^0 odgovara pozicija 1. Provjera da li je dekodiranje uspješno obavljeno postiže se dijeljenjem polinoma:

```
[ce,ost]=dij_bin_pol(xr(15:−1:1),parametri_bch(1,tip));
```

Realizacija funkcije dij_bin_pol je:

```
function [kol,ost]=dij_bin_pol(x1,x2)
m1=min(find(x1==1)); x1=x1(m1:length(x1));
m2=min(find(x2==1)); x2=x2(m2:length(x2));
if(length(x2)>length(x1))
    kol=0;
    ost=x1;
else
    kol1=[1,zeros(1,length(x1)−length(x2))];
    pro1=rem(conv(kol1,x2),2);
    kol2=pol_bin_sum(x1,pro1);
    mk=min(find(kol2==1));
    if(isempty(mk))
        ost=[];
        kol=kol1;
    else
        kol2=kol2(mk:length(kol2));
        [kolx,ost]=dij_bin_pol(kol2,x2);
        kol=pol_bin_sum(kol1,kolx);
    end
end
```

U slučaju da na kraju programa dobijemo nenulti ostatak pri dijeljenju ili da se dobijeni količnik razlikuje od polaznog polinoma, prijavljujemo pogrešku, a u suprotnom imamo uspješno obavljeno dekodiranje.

G. Prethodni primjer je više ilustrativan kako bi polaznici pratećeg kursa mogli pratiti tačnost svojih rješenja u dijelu izrade zadataka vezanih za BCH kodiranje i dekodiranje. U okviru MATLAB-ovog komunikacionog toolboxa postoji funkcija fec.bchenc kojom se mogu generisati parametri BCH koda. Funkcijom enc=fec.bchenc(15,5); dobijamo sve važne podatke za koder, u ovom slučaju to je BCH kod (15,3) (parametri funkcije su dužina riječi i broj informacionih bita). Kodiranje se obavlja funkcijom code=encode(enc,[1 1 0 1 1]');. Promijenimo sada tri bita code([3 9 11])=xor(code([3 9 11]),[1 1 1]'). Dekodiranje se obavlja obrnutom procedurom:

```
dec=fec.bchdec(15,5);
decode=decode(dec,code);
```

Dobijeni rezultat je jednak polaznoj kodnoj riječi:

```
decode'
    1     1     0     1     1
```

Preporučujemo studentima da izuče ove naredbe, kao i prateći *toolbox*, u kojem postoje mnoge korisne opcije u dijelu BCH kodiranja i kodne teorije.

POGLAVLJE VII

KONVOLUCIONI I TURBO-KODOVI

VII. KONVOLUCIONI I TURBO-KODOVI

Rad sa blok kodovima je relativno jednostavan. Informacioni biti se dijele u blokove, koji se zatim kodiraju dodavanjem redundandnih bita. Međutim, ovakva jednostavnost, posebno u fazi kodiranja, a kod mnogih kodova i u fazi dekodiranja, plaća se gubitkom performansi. Naime, poznato je da po II Šenonovoj teoremi za velike vrijednosti n i dati kanal kapaciteta C možemo dizajnirati kod sa kodnim odnosom $R \leq C$ koji daje vjerovatnoću greške u dekodiranju koja teži nuli. Međutim, takve dužine kodne riječi n nisu od praktičnog značaja pošto bi hardver bio skup, kašnjenje u dekodiranju neprihvatljivo itd. Stoga su određene i dostižne vjerovatnoće greške za fiksno n. Vjerovatnoće greške koje se postižu sa blok kodovima su znatno iznad dostižnih. Stoga smo prinuđeni ili da smanjujemo kodni odnos dodavanjem redundantnih bita, što prouzrokuje ekonomske gubitke zbog većih komunikacionih i memorijskih zahtjeva, ili da produžavamo n, što prouzrokuje gubitke zbog usložnjavanja hardvera. Razlog ovakvih performansi leži baš u bloku. Naime, blok izolovano posmatra određeni dio poruke, te ne postoji veza između jednog i drugog bloka, pa se biti iz različitih blokova ne mogu koristiti u procesu dekodiranja. Prvi odgovor na ovaj izazov su **konvolucioni kodovi**. Danas se intenzivno koriste kod digitalnog radija i videa, u mobilnim i satelitskim komunikacijama itd. Često se kombinuju i sa blok kodovima (u konkatenaciji, nadovezani jedni na druge). Način konstrukcije konvolucionih kodera, te osnove kodiranja biće obrazložene u Sekciji VII.1, dok su algoritmi dekodiranja prezentirani u Sekciji VII.2. Naredno unapređenje predstavljaju **turbo-kodovi**. Dajemo opis vrlina ovih sistema, s osnovnim principima kodiranja i dekodiranja. Turbo-kodovi su danas najbliži teorijskim limitima koje je uspostavila II Šenonova teorema. Postoji i još jedna alternativa, a to su **LDPC kodovi** (kodovi za kontrolu parnosti niske gustine – engl. *low-density parity-check codes*), koji će rudimentarno biti predstavljeni u narednom poglavlju.

VII.1 Konvolucioni kodovi

Logika konvolucionih kodova je nešto drugačija od one kod blok kodova. Kodna riječ se propušta kontinualno kroz koder, bez dijeljenja u blokove i kao takva se dekodira (ili, ako se dijeli u blokove, oni su znatno duži nego kod blok kodova). Na ovaj način u prijemnoj poruci postoji korelacija za kompletnu (ili veliku) dužinu kodne riječi, čime se nezavisnost između pojedinih blokova izbjegava. Dakle, bitovi koji mogu biti veoma razmaknuti u kodnoj riječi i primljenoj poruci mogu uticati na unapređenje procesa dekodiranja. Kod konvolucionih kodera posjedujemo više blokova za kodiranje (više manjih kodera), koji su po pravilu raspoređeni paralelno. Ovi koderi ne moraju svi da predstavljaju množenje sa prostim polinomima, kakav je slučaj bio kod blok kodova. Pored predmetnih kodova, informacioni biti mogu se propuštati direktno u kanal (direktno predstavljeni u kodnoj riječi), pa su u tom slučaju dobijeni kodovi **sistematski**. U suprotnom, imali bismo slučaj nesistematskih kodova. Pojedinačni koderi u okviru konvolucionih kodova mogu, a ne moraju, biti rekurzivni (sa povratnom spregom). Konvolucioni kodovi su i dalje najbolji u sistemima sa izuzetno malim kodnim odnosima (ili malim kapacitetom kanala), a često se za takve okolnosti koriste u kombinaciji sa blok kodovima, a posebno sa **Rid−Solomonovim** (engl. *Reed−Solomon*) **kodovima** (Poglavlje VIII.5).

Kodiranje konvolucionih kodova je trivijalan postupak, ništa složeniji od kodiranja blok kodova. Međutim, postupci dekodiranja, koji će biti obrađeni u narednoj sekciji, daleko su od jednostavnih. Najbolji dekodirajući postupak je **Viterbijev algoritam**. Pokazano je da se kod Viterbijevog algoritma gotovo iste performanse u smislu greške dekodiranja postižu kada je sekvenca izdijeljena na blokove od po nekoliko desetina, eventualno stotinjak bita, kao i kada se posmatra čitava kodna riječ. Veoma duge riječi prouzrokuju kašnjenja u procesu dekodiranja, usložnjavaju hardver, a kod Viterbijevog algoritma dovode do velikih memorijskih zahtjeva. Stoga se riječi i kod konvolucionih kodova dijele u manje blokove – kažemo da su **ogranične dužine** kako bismo izbjegli opisane probleme, ali ne nauštrb performansi koda (vjerovatnoće greške). Pojedinačni koderi u konvolucionim kodovima su u praksi kraći od onih kod blok kodova, te ne moraju svi biti zasnovani na prostom polinomu.

VII.1.1 Generisanje konvolucionog koda

Posmatrajmo sada jedan jednostavni blok kod sa generatorskim polinomom $\mathbf{g}(x)$:

$$\mathbf{c}(x)=\mathbf{i}(x)\mathbf{g}(x)=\left[\sum_{j=0}^{k-1} i_j x^j\right]\left[\sum_{l=0}^{n-k} g_l x^l\right]=\sum_{j=0}^{k-1}\sum_{l=0}^{n-k} g_l i_j x^{j+l}$$
$$=\sum_{j=0}^{n-1}\left[\sum_{l=0}^{\min(n-k,\,j)} g_l i_{j-l}\right]x^j.$$

Predmetna operacija je ista kao konvolucija diskretnog signala, pa se i kodovi kod kojih se kodne riječi dobijaju na ovaj način nazivaju konvolucionim! Ovo nam sugeriše činjenicu da su i blok kodovi suštinski konvolucioni ako bismo terminologiju posmatrali strogo na osnovu matematičke operacije koja se obavlja u koderu. Međutim, u teoriji kodova na ovom mjestu dolazi do napuštanja čistoće matematičkog formalizma.

Karakteristika konvolucionih kodova je da imamo više kodera. Stoga se generatorska matrica obično prikazuje u obliku polinoma koji predstavljaju navedene kodere. Uvedimo dvije generatorske matrice (primjeri preuzeti iz udžbenika [2,3]):

$$\mathbf{G}_1 = [x^2 + 1, x^2 + x + 1],$$

koja predstavlja konvolucioni kod sa dva generatorska polinoma, koji za svaki ulazni bit generiše dva izlazna (biće korišćen u više primjera);

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & x+1 \\ 0 & 1 & x \end{bmatrix},$$

koja za svaka dva ulazna informaciona bita generiše tri izlazna bita kodne riječi.

Osnovne karakteristike konvolucionih kodova su:

- memorija: $M = \max[\deg(g_{ij})]$ (deg označava stepen polinoma od engl. *degree*), koja predstavlja maksimalan red polinoma koji se pojavljuje u generatorskoj matrici (veoma bitna za konstrukciju kodera);
- dubina (ponekad se naziva ograničena dužina – *constraint length*): $N = M + 1$, koja pokazuje da aktuelni bit poruke zavisi od ulaznog bita i prethodnih M informacionih bita;
- kodni odnos: $R = k/n$ (gdje su $k \times n$ dimenzije matrice $\mathbf{G}$). Napominjemo da kada ograničimo dužinu kodne riječi, kodni odnos biće nešto umanjen, jer je ova vrijednost asimptotski data samo za beskonačno mnogo ulaznih bita, dakle, bez ograničenja, što će biti ilustrovano na primjerima.

Prethodna dva koda imaju respektivno sljedeće parametre: $M = 2$, $N = 3$, $R = 1/2$ i $M = 1$, $N = 2$, $R = 2/3$.

Kodiranje poruke može se prikazati putem predmetne kombinacije matričnog i polinomijalnog zapisa, kao proizvod:

$$\mathbf{c} = \mathbf{i}\mathbf{G}.$$

Ilustrujmo ovo množenje (kodiranje) na primjeru. Neka je informacioni polinom u slučaju prvoga koda $\mathbf{i}(x) = x^3 + x^2 + 1$. Množenjem dobijamo:

$$\mathbf{c} = \left[x^3 + x^2 + 1\right]\left[x^2 + 1, x^2 + x + 1\right] = \left[x^5 + x^4 + x^3 + 1, x^5 + x + 1\right].$$

Predmetna poruka se šalje u kanal tako što se propuštaju koeficijenti istog reda polinoma, počevši od najnižeg: (1,1), (0,1) (koeficijenti uz x^1), (0,0) (koeficijenti uz x^2), (1,0), (1,0), (1,1). Ovakva operacija se naziva konverzijom paralele (jer imamo dvije paralelne grane za dva kodera) u seriju (jer u kanalu možemo imati samo serijski raspored bita). U slučaju drugog konvolucionog koda, koji smo uzeli kao primjer, pretpostavimo da je na ulazu stiglo $\mathbf{i} = [x^2 + x, x^3 + 1]$. Rezultat se onda može zapisati kao:

$$\mathbf{c} = [x^2 + 1, x^3 + 1]\begin{bmatrix} 1 & 0 & 1+x \\ 0 & 1 & x \end{bmatrix} = [x^2 + 1, x^3 + 1, (x+1)(x^2 + 1) + x(x^3 + 1)] =$$
$$= [x^2 + 1, x^3 + 1, x^4 + x^3 + x^2 + 1].$$

Ponovo se izlaz može prikazati, nakon konverzije serije u paralelu, kao niz bita (počevši od bita uz koeficijente polinoma najmanjeg stepena ka onima većeg):

$$(1,1,1),(0,0,0),(1,0,1),(0,1,1),(0,0,1).$$

U zagradama smo, radi jasnoće, upisivali bite koji se nalaze uz koeficijente istog reda. Kodnu riječ možemo da prikažemo kao kombinaciju više (n) kodnih riječi $\mathbf{c} = [c_0(x), c_1(x), \ldots c_{n-1}(x)]$ koje su odziv kodera na k ulaznih sekvenci $\mathbf{i} = [i_0(x), i_1(x), \ldots, i_{k-1}(x)]$. Napominjemo da u primjeru nismo uveli ograničenja na dužinu predmetnih sekvenci (ni informacionih ni kodnih riječi). Svaki od kodnih polinoma može se zapisati kao $\mathbf{c}_j(x) = c_{j0} + c_{j1}x + \cdots$, pa se kodna riječ, nakon konverzije paralele u seriju, može zapisati kao:

$$\mathbf{c} = [c_{00}c_{10}\ldots c_{n-1,0}; c_{01}c_{11}\ldots c_{n-1,1}; c_{02}c_{12}\ldots c_{n-1,2}; \ldots]$$

Generatorski polinom prvog uvedenog konvolucionog koda možemo zapisati i kao:

$$\mathbf{G}(x) = [x^2 + 1, x^2 + x + 1] = [1,1] + [0,1]x + [1,1]x^2$$
$$= \sum_{i=0}^{M} G_i x^i$$

gdje je $G_0 = [1,1]$, $G_1 = [0,1]$ i $G_2 = [1,1]$. Pored uvedenih načina prikaza, generatorska matrica se možda najpogodnije prikazuje u tzv. skalarnoj formi:

$$\mathbf{G} = \begin{bmatrix} G_0 & G_1 & G_2 & \ldots & G_M & 0 & 0 & \ldots \\ 0 & G_0 & G_1 & \ldots & G_{M-1} & G_M & 0 & \ldots \\ 0 & 0 & G_0 & \ldots & G_{M-2} & G_{M-1} & G_M & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix},$$

koja za prvi uvedeni konvolucioni kod ima oblik:

$$\mathbf{G}=\begin{bmatrix} 11 & 01 & 11 & & & & \\ & 11 & 01 & 11 & & & \\ & & 11 & 01 & 11 & & \\ & & & 11 & 01 & 11 & \\ & & & & & \downarrow & \rightarrow \end{bmatrix}.$$

Prilikom prikaza ove matrice, izostavili smo sve nulte elemente. U predmetnoj matrici postavili smo strelice da ukažu na pravce širenja, vezano za broj informacionih bita. Treba reći da dimenzije ove matrice (pod uslovom da se ne uvedu neka ograničenja na broj informacionih bita, a o čemu će u nastavku biti riječi) korespondiraju broju informacionih bita. Kodirajmo informaciju, datu preko polinoma $\mathbf{i}=[x^3+x+1]$, putem ove generatorske matrice:

$$\mathbf{c}=[1\,1\,0\,1]\begin{bmatrix} 11 & 01 & 11 & & & \\ & 11 & 01 & 11 & & \\ & & 11 & 01 & 11 & \\ & & & 11 & 01 & 11 \end{bmatrix}=[111010000111].$$

Uočite da smo vektor koji predstavlja informacione bite prikazali od bita uz koeficijent polinoma najmanjeg stepena ka onima uz koeficijente većeg stepena. Provjerite rezultat postupka primjenom množenja korespondentnih polinoma. Kao jedna vrlina ovog postupka slijedi da nije potrebna konverzija paralele u seriju. Ovakvo matrično množenje se rijetko provodi iz više razloga, među kojima je jedan od bitnijih dimenzija matrica. Stoga, kako će kasnije biti pokazano, konverzija paralele u seriju češće se provodi u praksi, ali ovakvo matrično množenje može ponekad da bude uputno za razumijevanje funkcionisanja postupka konvolucionog kodiranja.

Sličan postupak se može provesti i kod drugog konvolucionog koda, čiji se generatorski polinom može prikazati u obliku:

$$\mathbf{G}(x)=\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}+\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}x,$$

gdje je $G_0=\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ i $G_1=\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$. Sada se skalarna generatorska matrica ovog koda može zapisati kao (uz izostavljanje nultih elemenata i uz saznanje da dimenzije ove generatorske matrice mogu biti praktično beskonačne u zavisnosti od broja bita informacione riječi):

$$\mathbf{G}=\begin{bmatrix} 101 & 001 & & \\ 010 & 001 & & \\ & 101 & 001 & \\ & 010 & 001 & \\ & & \downarrow & \rightarrow \end{bmatrix}.$$

Uzmimo primjer da je sa ovom matricom potrebno kodirati informaciju zapisanu preko polinoma $\mathbf{i}_0(x)=x^2+x$ i $\mathbf{i}_1(x)=x^3+1$ ili kao $\mathbf{i}=[x^2+x,x^3+1]$. Množenjem ove informacione riječi sa generatoskim polinomom dobijamo, na osnovu prethodno opisanog postupka, kodni polinom s elementima:

$$\begin{aligned}\mathbf{c}&=[x^2+x,x^3+1,(x^2+x)(x+1)+(x^3+1)x]=\\&=[x^2+x,x^3+1,x^4+x^3],\end{aligned}$$

odnosno, nakon konverzije paralele u seriju (od koeficijenta uz stepene najnižeg reda pa naviše):

$$\mathbf{c}=[010;100;100;011;001].$$

Trojke simbola su samo za potrebe lakšeg praćenja razdvojene sa ;. U skalarnoj formi, informacija se sada može prikazati (polazeći od koeficijenta uz stepen najmanje težine i provodeći konverziju paralele u seriju) kao [01101001]. Pomnožimo predmetnu informaciju sa skalarnom generatorskom matricom odgovarajućih dimenzija:

$$\mathbf{c}=[01101001]\begin{bmatrix} 101 & 001 & & & & \\ 010 & 001 & & & & \\ & 101 & 001 & & & \\ & 010 & 001 & & & \\ & & & 101 & 001 & \\ & & & 010 & 001 & \\ & & & & 101 & 001 \\ & & & & 010 & 001 \end{bmatrix}=[010;100;100;011;001].$$

Za sada nije diskutovan izbor odgovarajućih generatorskih polinoma kod konvolucionih kodova. Ta materija nije jednostavna, te se obično i ne izlaže na ovom nivou materijala. Najčešće se do dobrih generatorskih polinoma dolazi kompjuterskim simulacijama, te se ovi kodovi prikazuju tabelarno ili na druge pogodne načine premda postoje neka osnovna pravila kako se izbor može vršiti. Najčešće, to su prosti polinomi, ali, kao što smo upravo vidjeli, koristimo i polinome koji nisu prosti, kao što je, na primjer, bio x^2+1. Ipak, nećemo objašnjavati detalje vezane

za izbor generatorskih polinoma osim u neophodnoj formi. Dosta je uobičajeno da se izbor polinoma vrši na osnovu simulacija, u skladu sa kanalom za koji se kod primjenjuje.

VII.1.2 Konvolucioni kodovi s ograničenjem

Pretpostavili smo za sada da u konvolucionom kodu ne postoji ograničenje, odnosno da možemo da propustimo beskonačnu povorku bitova i da ih kodiramo bit za bitom. Vrlina ovakvog postupka je u tome da smo u nekom obliku bliži zahtjevima II Šenonove teoreme da bi kodna riječ trebala da bude što je moguće duža, kako bi se sa što je moguće manjim brojem redundantnih bita obezbijedio prenos poruka bez greške. Međutim, zbog raznih praktičnih ograničenja, prinuđeni smo da ograničimo konvolucione kodove. Naime, umjesto da prenesemo sve bite poruke, bićemo prinuđeni da „odsiječemo" dio poruke (informacionih bita), te da taj dio kodiramo i pošaljemo u kanal, a zatim da uzmemo drugi (uslovno govoreći) blok bita informacije, kodiramo ga i pošaljemo u kanal. Osnovni razlog zbog kojeg provodimo ovu operaciju je olakšavanje dekodiranja. Ako bi poruka bila kodirana u cjelosti, posebno ako je poruka veoma velike dužine (jedna slika ili jedna video-sekvenca može biti velika nekoliko miliona ili milijardi bitova), dekoder bi suštinski morao da prikupi sve te bite i na osnovu njih da donese odluku. Kao što ćemo uskoro vidjeti, proces dekodiranja je daleko od jednostavnog, pa je zgodno smanjiti dužinu kodne riječi kako bi se redukovali potrebni hardverski resursi (i računski i memorijski) za dekodiranja. Postavlja se pitanje: u kojoj mjeri ovo ugrožava zahtjeve koje smo postavili pred konvolucione kodove, pošto se na neki način vraćamo blizu situacije koju smo imali kod blok kodova? Na sreću, naučnici su pokazali (uglavnom simulacijama) da je gubitak ovog postupka relativno mali ako je ograničenje razumno veliko (što često ne prelazi nekoliko desetina ili eventualno stotina bita).

Pretpostavimo da smo ograničili informacionu riječ na L bita, odnosno da je svaki od informacionih polinoma ograničen na $L-1$ stepen:

$$\deg[i_j(x)] \le L-1, j=0,\ 1,\ ...,\ k-1,$$

U ovom slučaju, kodni polinomi $\mathbf{c}=[c_0(x),c_1(x),...,c_{n-1}(x)]$ imaju stepen $M+L-1$ jer su generatorski polinomi stepena M. Informaciju u ovom slučaju čini kL bita, dok je $n(M+L)$ bita u kodnoj riječi. Čak i bez neke velike nepreciznosti, ovakav kod bi se mogao označiti u obliku blok koda sa parametrima $(n(M+L),kL)$. Kodni odnos ovoga koda je:

$$R_L=\frac{kL}{n(M+L)}=R\left(1-\frac{M}{M+L}\right),$$

gdje $R = k/n$ predstavlja kodni odnos koda bez ograničenja. Vidimo da uvođenje ograničenja donekle umanjuje kodni odnos, ali ako je ograničenje relativno veliko u odnosu na red polinoma M, to umanjenje ne mora da bude znatno.

Generisanje konvolucionog koda sa ograničenjem može se prikazati putem izraza oblika:

$$\mathbf{c} = \mathbf{i}\mathbf{G}_L,$$

gdje je $\mathbf{G}_L$ generatorska matrica koda s ograničenjem:

$$\mathbf{G}_L = \underbrace{\left.\begin{bmatrix} G_0 & G_1 & G_2 & \dots & G_M & 0 & 0 \dots 0 \\ 0 & G_0 & G_1 & \dots & G_{M-1} & G_M & 0 \dots 0 \\ 0 & 0 & G_0 & \dots & G_{M-2} & G_{M-1} & G_M \dots 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \dots \dots \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots \dots G_M \end{bmatrix}\right\} L - \text{blokova}.}_{M+L-\text{blokova}}$$

Kod prvog konvolucionog koda za ograničenje $L = 6$ generatorska matrica postaje (uz izostavljene nulte elemente):

$$\mathbf{G}_6 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & & & & & & & & & & \\ & & 0 & 1 & 0 & 1 & 1 & 1 & & & & & & & & \\ & & & & 1 & 1 & 0 & 1 & 1 & 1 & & & & & & \\ & & & & & & 1 & 1 & 0 & 1 & 1 & 1 & & & \\ & & & & & & & & 1 & 1 & 0 & 1 & 1 & 1 & \\ & & & & & & & & & & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix},$$

dok je kod drugog uvedenog konvolucionog koda, na primjer, za ograničenje $L = 2$, generatorska matrica:

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & & & \\ 0 & 1 & 0 & 0 & 0 & 1 & & & \\ & & & 1 & 0 & 1 & 0 & 0 & 1 \\ & & & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Kod ovog pristupa (jedino mogućeg u praksi) kodira se L bita, a kao što vidimo iz matrice, ona ima $M+L$ blokova, pa se stoga na kraj vektora koji predstavlja informacione bite dodaje M nula kako bi se postiglo zadovoljenje dimenzija matrične relacije. Ujedno, kao što će biti jasnije iz naredne sekcije, na ovaj način se „resetuje“ (postavlja na početno stanje) kodersko kolo kako bi bilo spremno za kodiranje naredne serije od L informacionih bita.

VII.1.3 Realizacija kodera

Posmatrajmo prvi uvedeni konvolucioni kod. Suštinski, sastoji se od dva blok koda sa generatorskim polinomima $\mathbf{g}_1(x) = x^2 + 1$ i $\mathbf{g}_2(x) = x^2 + x + 1$. Ova dva blok koda predstavljena su na Slici VII.1 (uz napomenu da se ovdje poruka ne dijeli u relativno kratke serije bita, nego se šalje veći broj bita, vodeći računa o ograničenju). Vidimo da su u ovoj realizaciji neophodne četiri ćelije pomjeračkog registra, kao i tri ekskluzivno ili kola. Međutim, pomjerački registar ima samo namjenu da zakasni pojedine bite, te se ova dva kola mogu kombinovati kako je prikazano na Slici VII.2, gdje vidimo da su za realizaciju potrebne dvije ćelije pomjeračkog registra sa tri ekskluzivno ili kola. Naravno, podaci se kroz komunikacioni kanal moraju slati „jednom linijom“, pa se na kraju ovog kola nalazi sistem koji konvertuje paralelne bite u serijsku sekvencu, kako je to prethodno već objašnjeno (prvi bit iz gornje linije, praćen prvim bitom donje linije, zatim par drugih bita, par trećih bita itd.). Sistem upotpunjuje logika koja dozvoljava da se L bita prenese u koder, a zatim se propusti još onoliko nula koliko ima ćelija u registru da bi se koder pripremio za kodiranje narednih L informacionih bita.

Kao što se može uočiti, predmetni koderi su veoma jednostavni, odnosno složenost konvolucionih kodova nije u postupku kodiranja, već je u dekodiranju. Za realizaciju složenijih sistema, kod kojih se, na primjer, neki od kodiranih bita mogu izostaviti kako bi se povećao kodni odnos u povoljnim uslovima komuniciranja, potrebno je sistemima, prikazanim na Slici VII.1 i Slici VII.2, dodati odgovarajuću logiku. Konvolucioni kodovi s ograničenjima zahtijevaju odgovarajuća logička kola jer je potrebno implementirati prekidač koji određuje koliko se bita poruke propušta i koji zatim dodaje onoliko nula koliko je potrebno da se kodersko kolo vrati u inicijalno stanje kako bi se pripremili za kodiranje naredne grupe bita.

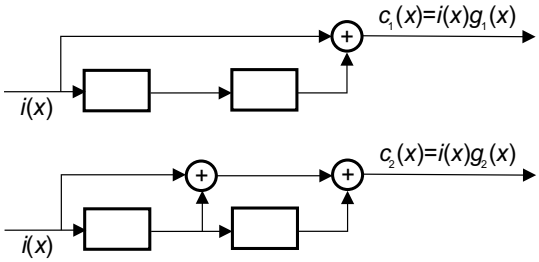
VII.1.4 Grafički prikaz procesa kodiranja

Prije nego što pređemo na izuzetno složenu problematiku vezanu za dekodiranje konvolucionih kodova, moramo da se upoznamo sa načinom na koji se kodiranje ovakvih kodova može vizuelizovati. Razlog za ovu digresiju u izlaganju je u činjenici da se zapravo predmetna vizuelizacija koristi i kod dekodiranja kodova, to jest da je bez vizuelizacije stanja u koderu i dekoder nemoguće razumjeti, a često ni implementirati proces dekodiranja.

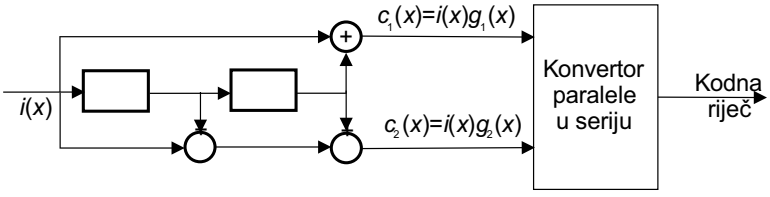
Srećna okolnost je ta što smo metodologije za vizuelizaciju procesa kodiranja i dekodiranja konvolucionih kodova već učili u drugom poglavlju, kod Markovljevih sistema. Dakle, dva najvažnija postupka za vizuelizaciju kodiranja i dekodiranja kod konvolucionih kodova jesu graf stanja i trelis dijagram. Trelis je toliko bitan i u suštini pregledan način vizuelizacije kod ovih kodova da su ranije (pogrešno)

ovi kodovi nazivali trelis kodovima, te da su odgovarajuće komunikaciono-modulacione tehnike nazivane **trelis kodnim modulacijama**.

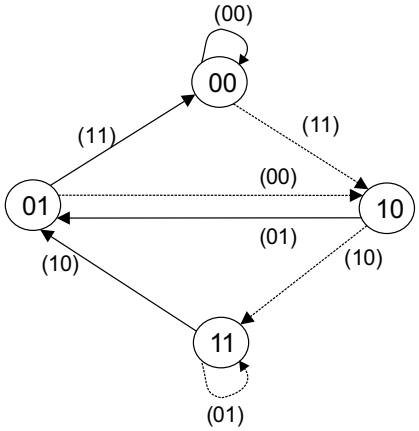
Posmatrajmo primjer prvog konvolucionog koda koji je korišćen u našim primjerima, a čiji je koder prikazan na Slici VII.1 i Slici VII.2. Pod stanjima kodera podrazumijevamo ono što je upisano u ćelije kodera, odnosno kombinacije 00, 01, 10 i 11. Ovdje se moramo dogovoriti o konvenciji kojim redosljedom da prikazujemo bite u stanju. Premda nije najčešće zastupljeno u literaturi, ovdje ćemo se držati konvencije koju smo imali kada smo u drugom poglavlju radili sa Markovljevim sistemima: stanja smo obilježavali tako što smo bite stanja prikazivali u obrnutom redosljedu od redosljeda pojavljivanja, odnosno prvi bit u stanju je bit koji neposredno prethodi, praćen bitima koji su njemu prethodili. Na Slici VII.3 i Slici VII.4 prikazan je graf prelaza između pojedinih stanja u koderu, kao i odgovarajući trelis dijagram. Slika VII.3 je slična Slici II.5, koja je korišćena za ilustraciju Markovljevog binarnog sistema drugog reda. Na ovom dijagramu nisu naglašene uslovne vjerovatnoće prelaza među pojedinim stanjima, već nam je bilo važnije da naglasimo simbole koje, nakon procesa kodiranja, šaljemo u kanal (navedene u zagradama iznad pojedinih linija). Tip linije predstavlja ulazni simbol koji treba da dekodiramo u procesu. Trelis, međutim, ima jednu bitnu razliku u odnosu na onaj koji je dat u Poglavlju II. Naime, kako počinjemo iz stanja 00, u prva dva takta nemamo sva stanja, već samo ona koja su moguća (prvo 00, a u narednom koraku, ako se šalje 0, prelazimo u stanje 00, dok ako se šalje 1, prelazimo u stanje 10). Već smo vidjeli da se konvolucioni kodovi najčešće implementiraju s ograničenjem, pa nakon slanja informacionih bita moramo vratiti



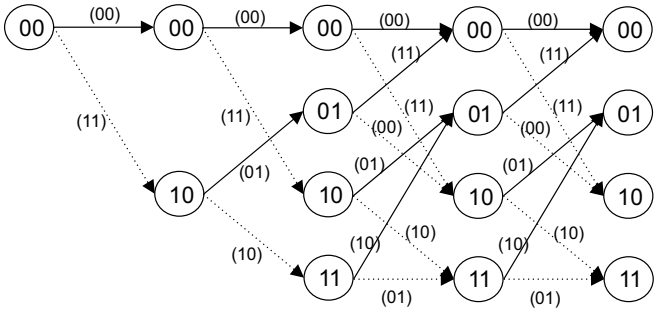
Slika VII.1. Konvolucioni koder sa generatorskim polinomima $\mathbf{G} = [x^2 + 1, x^2 + x + 1]$



Slika VII.2. Realizacija kodera sa manjim brojem ćelija pomjeračkog registra



Slika VII.3. Graf tranzicije za konvolucioni kod generisan koderima sa Slike VII.1 i Slike VII.2



Slika VII.4. Trelis dijagram za konvolucioni kod generisan koderima sa Slike VII.1 i Slike VII.2

ćelije kodera u početno stanje 00. Dakle, posljednja dva bita nisu informacioni, već služe za resetovanje ćelija kodera. Pretpostavimo da je $L = 6$ i dobijamo trelis dijagram prikazan na Slici VII.5.

Prije nego se pređe na objašnjenje algoritama koji se koriste za dekodiranje konvolucionih kodova, vrijedi ukazati još na jednu činjenicu. Koderi sa Slike VII.1 i VII.2 predstavljaju množenje odgovarajućih informacionih polinoma sa generatorskim polinomima. Postoji, naravno, mogućnost da se konvolucioni kodovi realizuju i putem kodera sa povratnom spregom. Radi jednostavnosti, za sada se držimo kodova dobijenih množenjem polinoma.

VII.2 Dekodiranje konvolucionih kodova

Postoje tri postupka za dekodiranje konvolucionih kodova: majoritetna logika, sekvencijalno dekodiranje i Viterbijev algoritam. Njihova složenost i polje primjene su različiti, što će biti objašnjeno tokom izlaganja.

VII.2.1 Majoritetna logika

Radi ilustracije, posmatrajmo koder, prikazan na Slici VII.2, u procesu kodiranja trobitne informacione poruke $\mathbf{i}(x) = i_2x^2 + i_1x + i_0$. U dvije grane kodera dobijamo sljedeće kodne riječi:

$$\mathbf{c}_1(x) = (i_2x^2 + i_1x + i_0)(x^2 + 1) = i_2x^4 + i_1x^3 + i_2x^2 + i_0x^2 + i_1x + i_0$$
$$\mathbf{c}_2(x) = (i_2x^2 + i_1x + i_0)(x^2 + x + 1) = i_2x^4 + (i_2 + i_1)x^3 + (i_2 + i_1 + i_0)x^2 + (i_1 + i_0)x + i_0.$$

Dakle, dobijamo dvije kodne riječi sa kodnim simbolima koje možemo da označimo kao: $\{c_{14}, c_{13}, c_{12}, c_{11}, c_{10}\}$ i $\{c_{24}, c_{23}, c_{22}, c_{21}, c_{20}\}$. Možemo da izrazimo i_2 direktno iz kodne riječi, kao:

$$i_2 = c_{14}$$
$$i_2 = c_{24},$$

ali se informacioni bit i_2 može izraziti i kao:

$$i_2 = c_{23} + c_{13} = i_2 + i_1 + i_1.$$

Dakle, odluku o bitu i_2 možemo donijeti na osnovu c_{14}, c_{24} i $c_{23} + c_{13}$ i, kako imamo više mogućnosti u uslovima pojave grešaka u kanalu, odluku donosimo majoritetnom (većinskom) logikom, odnosno većinom glasova! Kada dekodiramo neki bit, to možemo dalje da koristimo za dekodiranje ostalih bita. Tako, na primjer, bit i_1 možemo da dekodiramo na osnovu tri izraza:

$$i_1 = c_{13}$$
$$i_1 = c_{11}$$
$$i_1 = c_{23} + i_2.$$

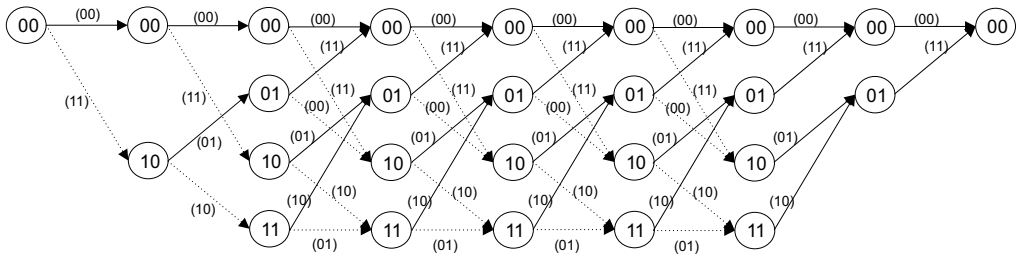
Ponovo imamo tri situacije, pa se na osnovu majoritetne logike donosi odluka o bitu i_1. U praksi postoje i situacije kada je broj izraza na osnovu kojih se vrši provjera veći, pa čak i sa parnim brojem provjera na nekim bitima. U slučaju parnog broja provjera i „neriješenog skora“, uzima se proizvoljan bit kao dekodiran, a ujedno to ukazuje i na veliki broj grešaka koje postoje u kanalu, te o mogućnosti da sistem dekodiranja nije adekvatan situaciji (vjerovatnoći grešaka) u kanalu.

Dakle, osnovna ideja kod majoritetne logike je da izrazimo informacione bite (ili grešku koja se može dogoditi u kanalu) preko više izraza i da na osnovu većine glasova donesemo odluku o tome koji je bit poslat. Zatim, taj bit koristimo u dekodiranju ostalih bita, podrazumijevajući da je tačan. I kod ovog postupka često posežemo za formiranjem sindromskih bita ili sindromskih izraza (kao što je urađeno u datom primjeru). Ako se neki od informacionih bita pojavljuje u

izrazima za sve sindrome, onda kažemo da je kod **ortogonalan** u odnosu na taj informacioni bit (ili bit pogreške, u zavisnosti od toga kako su sindromski izrazi formirani). Praksa pokazuje da su za primjenu majoritetne logike pogodni samo oni konvolucioni kodovi koji se mogu efektivno ortogonalizovati. Pošto postupak ortogonalizacije nije uvijek moguć, te pošto majoritetna logika ne daje kod konvolucionih kodova rezultate koji su kompetitivni s drugim postupcima dekodiranja (u smislu vjerovatnoće greške), ovaj postupak se danas dosta rijetko koristi za dekodiranje konvolucionih kodova, premda nije generalno nepopularan kao dio drugih postupaka u teoriji kodova.

VII.2.2 Viterbijev algoritam

Najpogodnije je da razmotrimo oba preostala postupka dekodiranja konvolucionih kodova (Viterbijev i sekvencijalni) odjednom jer koriste u osnovi isti alat − trelis. Cilj oba algoritma je da se pronađe putanja kroz trelis koja ima najmanje Hemingovo (ili neko drugo) rastojanje u odnosu na kodnu riječ koja je primljena. Pošto je Viterbijev algoritam kvalitetniji postupak, jedino će on biti demonstriran za ovaj tip algoritma, dok će sekvencijalno dekodiranje biti kratko protumačeno. Osnovna mana Viterbijevog algoritma dugo vremena je bila memorijska složenost. Međutim, kako današnji sistemi raspolažu značajnim memorijskim resursima, ovaj problem je uglavnom prevaziđen ili relativizovan. Viterbijev postupak koristi se danas u brojnim oblastima nauke, a inicijalno ga je razvio Endrju Viterbi (engl. *Andrew Viterbi*) za potrebe dekodiranja konvolucionih kodova. Interesantna je činjenica da je ovu ideju dobio tokom gledanja jedne igre, koja se izvodi tokom religijske svečanosti (jevrejskog praznika Purim), u kojoj učestvuje više pokoljenja (što bi Njegoš rekao „s unučađu đedovi igraju, po tri pasa vrte se u kolo“). Viterbi je, gledajući igru u kojoj su učestvale razne generacije, došao na ideju da poveže „kolo“ čvorova u trelisu iz različitih trenutaka i da razvije revolucionarni algoritam dekodiranja, koji omogućava da se formira najbolja putanja kroz trelis na osnovu putanja koje su postojale do prethodnih čvorova. Ponovo pozivamo u pomoć primjer kodera čiji je trelis dijagram prikazan na Slici VII.5.



Slika VII.5. Trelis dijagram za konvolucioni kod generisan koderima s ograničenjem $L = 6$ prikazanim na Slici VII.1 i Slici VII.2

Riječ je o koderu sa samo dvije ćelije u registru (Slika VII.2), tako da trelis ima samo četiri stanja. Kako započinjemo iz stanja 00, u prvom koraku možemo preći samo u dva stanja (dvije putanje), iz drugog koraka (prouzrokovanog drugim bitom) možemo preći u sva četiri stanja (četiri putanje), a od te tačke nastavljamo sa punim trelisom od po četiri stanja (osam putanja). U opštem slučaju, kada konvolucioni koder ima r ćelija u registru, to znači da ima 2^r stanja. Prvih r bita u praznom koderu označava 2^r mogućih putanja. Nakon toga, iz svakog stanja može se preći u dva nova stanja, tako da u svakom koraku imamo 2^{r+1} mogućih putanja. Ovo pravilo važi za narednih $L-r$ bita. Dakle, ukupan broj putanja u L bita je 2^r (za prvih r bita) $\times\, 2^{(L-r)(r+1)}$ (za putanje u $L-r$ bita punog trelisa). Konačno, u posljednjih r bita upisuju se nule kako bi se koder pripremio za novu nezavisnu sekvencu, tako da su preostalih 2^r putanja determinisane prethodnim stanjima (pod uslovom da nema grešaka u sistemu). Kod sistema sa 2^r stanja ukupan broj putanja je 2^{Lr+L-r^2}, što može biti ogroman broj. Na primjer, za relativno uobičajen scenarij od $L=20$ (ili više) i $r=3$ (a često je i više) broj putanja je $2^{71}\approx 2.36\cdot 10^{21}$. Naravno, konvolucioni kodovi su konstruisani za slučaj kada ima grešaka u sistemu. To znači da, umjesto da primimo neku od dozvoljenih putanja u sistemu, mi primamo riječ koja ne korespondira nijednoj od putanja u trelisu. Trebalo bi da rezultat dekodirajućeg algoritma bude ona ispravna riječ (putanja) koja se najmanje razlikuje od primljene riječi. Uzmimo da se razlika mjeri Hemingovom distancom (mada ćemo kasnije vidjeti da te razlike mogu da budu i drugačije). Očigledno je da je direktna pretraga u skupu od $2.36\cdot 10^{21}$ mogućih kodnih riječi (mogućih putanja u trelisu) potpuno neprihvatljiva. Stoga se morao iznaći novi algoritam kojim bi se ovaj postupak učinio efikasnim.

Viterbijev postupak je iterativna procedura koja se ponavlja za svaki trenutak u trelisu. Naime, ako posmatramo sve čvorove u trelisu, u nekom trenutku do svakog od njih smo mogli da stignemo sa više putanja (ne računajući nekoliko početnih trenutaka). Od svih tih putanja, jedna je najbolja. Dakle, dovoljno je pamtiti samo tu putanju među svim putanjama kojima se moglo doći do datog čvora. Ova putanja se naziva **parcijalnom najboljom putanjom**. Broj parcijalnih najboljih putanja jednak je broju čvorova u datom trenutku. U sljedećem trenutku određujemo najbolje parcijalne putanje do narednih čvorova. Za dijagram sa Slike VII.5 u svaki čvor se stiže sa dvije putanje iz prethodnih čvorova. Kod nas konkretno, te putanje do novih čvorova su konkatenacije između parcijalnih najboljih puteva do prethodnih čvorova i putanje između dva čvora. Dakle, ponovo u narednom trenutku pamtimo najbolje parcijalne putanje, izborom između dvije moguće, kao nadovezivanje (konkatenaciju) najboljih parcijalnih putanja do prethodnih čvorova i putanja između dva čvora. Pored putanja moramo pamtiti i metrike, odnosno rastojanja ili kriterijume po kojima mjerimo kvalitet pojedine putanje. Najjednostavniji način je da se to radi putem Hemingove distance.

U prvih r koraka imamo 2^r putanja koje inicijalno treba sve zapamtiti. U narednim koracima punog trelisa imamo 2^r stanja, povezanih sa 2^r stanja preko 2^{r+1} putanja. Dakle, u svakom koraku poredimo 2^{r+1} putanja. Imamo $L-r$ koraka punog trelisa i na kraju imamo još r koraka u kojima dolazi do redukcije trelisa kao stanja sa svim nulama. U svakom od tih koraka imamo po polovinu putanja manje u odnosu na prethodni korak (u našem slučaju sa Slike VII.5, sa 8 spadamo na 4, pa na 2), tako da za redukovani dio trelisa treba provjeriti ukupno $2^{r+1}-2$ putanja. Imamo, dakle, ukupan broj putanja koje treba porediti u ovom algoritmu:

$$2^r(\text{putanje početnog trelisa}) + (L-r)2^{r+1}\ (\text{putanja punog trelisa}) + 2^{r+1} - 2\ (\text{na kraju trelisa}) \;=\; 2^r[1+2(L-r)+2] - 2 \approx 2^{r+1}(L-r).$$

Za primjer sa $L=20$ i $r=3$ potrebno je provjeriti (i porediti) samo 270 putanja, što je približno 10^{19} puta manje nego da smo provjeravali svaku pojedinačnu moguću putanju u trelisu.

Ovakav rad ne dolazi bez neke mane. Naime, osnovni problem kod Viterbijevog algoritma su memorijski zahtjevi. Na primjer, u koraku L algoritma potrebno je pamtiti 2^r parcijalnih najboljih putanja dužine L. Nekada je ovo bio ograničavajući faktor, ali, pošto memorijski zahtjevi algoritma rastu linearno sa dužinom L, danas se ovo smatra prihvatljivom prostornom složenošću Viterbijevog algoritma.

Primjer VII.1. Posmatrajmo koder sa Slike VII.5. Pretpostavimo da treba da pošaljemo 6 bita 101011 i da su se u rezultujućoj kodnoj riječi greške desile na pozicijama 2 i 7. Izvršiti dekodiranje kodne riječi putem Viterbijevog algoritma.

Rješenje: Ispravna kodna riječ je: 1101000100101011. Primljena kodna riječ je onda 1**0**0100**1**100101011, gdje smo vizuelno izdvojili bite kod kojih se pojavila pogreška. Posmatrajmo sada trelis s ovim kodom. Iz stanja 00 imamo dvije putanje koje vode ka stanjima 00 i 10, obje imaju distancu 1 do primljene kodne riječi (primili smo 10, a putanje su sa simbolima 00 i 11). Obje ove putanje se pamte. Zatim imamo četiri putanje ka stanjima 00 (Hemingova težina 2 u odnosu na primljenu riječ), 01 (Hemingova težina 1), 10 (Hemingova težina 2) i 11 (Hemingova težina 3). Pamtimo sve četiri putanje. Nakon toga imamo primljena dva bita 00. U stanje 00 se može pristići preko dvije putanje, one iz 00 (Hemingova težina 2) i one iz stanja 01 (Hemingova težina 3). Ovdje pamtimo prvu putanju, sa Hemingovom težinom 2 (0-0-0). U stanje 01 može se doći iz stanja 10 (Hemingova težina 3) i 11 (Hemingova težina 4). Ponovo pamtimo prvu putanju, sa težinom 3 (0-1-0). U stanje 10 može se doći iz stanja 00 (težina 4) i stanja 01 (težina 1). Pamtimo drugu putanju (1-0-1), težine 1. Konačno, u trećem koraku se može doći u stanje 11 preko stanja 10 (težina 3) i stanja 11 (težina 4). Pamtimo samo prvu putanju, težine 3 (0-1-1). Dakle, poslije tri koraka imamo četiri parcijalna najbolja puta:

U stanje 00 težine 2 0-0-0

U stanje 01 težine 3 0-1-0

U stanje 10 težine 1 1-0-1

U stanje 11 težine 3 0-1-1.

U četvrtom koraku možemo do stanja 00 doći preko parcijalnog najboljeg puta do stanja 00 na koje se nadoveže 0 s ukupnom težinom 3, odnosno putanjom do stanja 01 na koje se nadoveže nula s težinom 3. Možemo izabrati bilo koju od putanja jer imaju iste Hemingove težine. U stanje 01 možemo doći ponovo preko dvije putanje, one od 10, sa dodatkom 0, težine 2 i one iz 11, ukupne težine 4. Biramo prvu putanju 1-0-1-0, težine 2. U stanje 10 možemo doći iz stanja 00, težine 2 i stanja 01, težine 5. Biramo, naravno, prvu putanju 0-0-0-1, težine 2. Konačno, u stanje 11 možemo doći iz stanja 10, težine 2 i stanja 11, težine 4. Ponovo biramo prvu putanju 1-0-1-1, težine 2. Dakle, parcijalni najbolji putevi do četvrtog koraka su:

U stanje 00 težine 3 0-0-0-0 (ili 0-1-0-0) biramo proizvoljnu

U stanje 01 težine 2 1-0-1-0

U stanje 10 težine 2 0-0-0-1

U stanje 11 težine 2 1-0-1-1.

Nastavljamo dalje. U petom koraku ponovo imamo puni trelis, tako da se do čvora 00 može doći iz stanja 00 (težine 3) i iz stanja 01 (težine 4). Biramo prvu putanju, gdje na prethodnu putanju nadovezujemo 0. U stanje 01 možemo doći iz stanja 10 (težine 3) i stanja 11 (isto težine 3). Ponovo imamo nedoumicu koju od mogućih parcijalnih najboljih putanja odabrati, ali su sa stanovišta dekodiranja ravnopravne. U stanje 10 možemo doći iz stanja 00 (težina 5) i stanja 01 (težina 2). Biramo drugu putanju, manje težine. Konačno, u stanje 11 možemo doći iz stanja 10 (težine 3) i stanja 11 (isto težine 3). Da sublimiramo stanje poslije pet koraka:

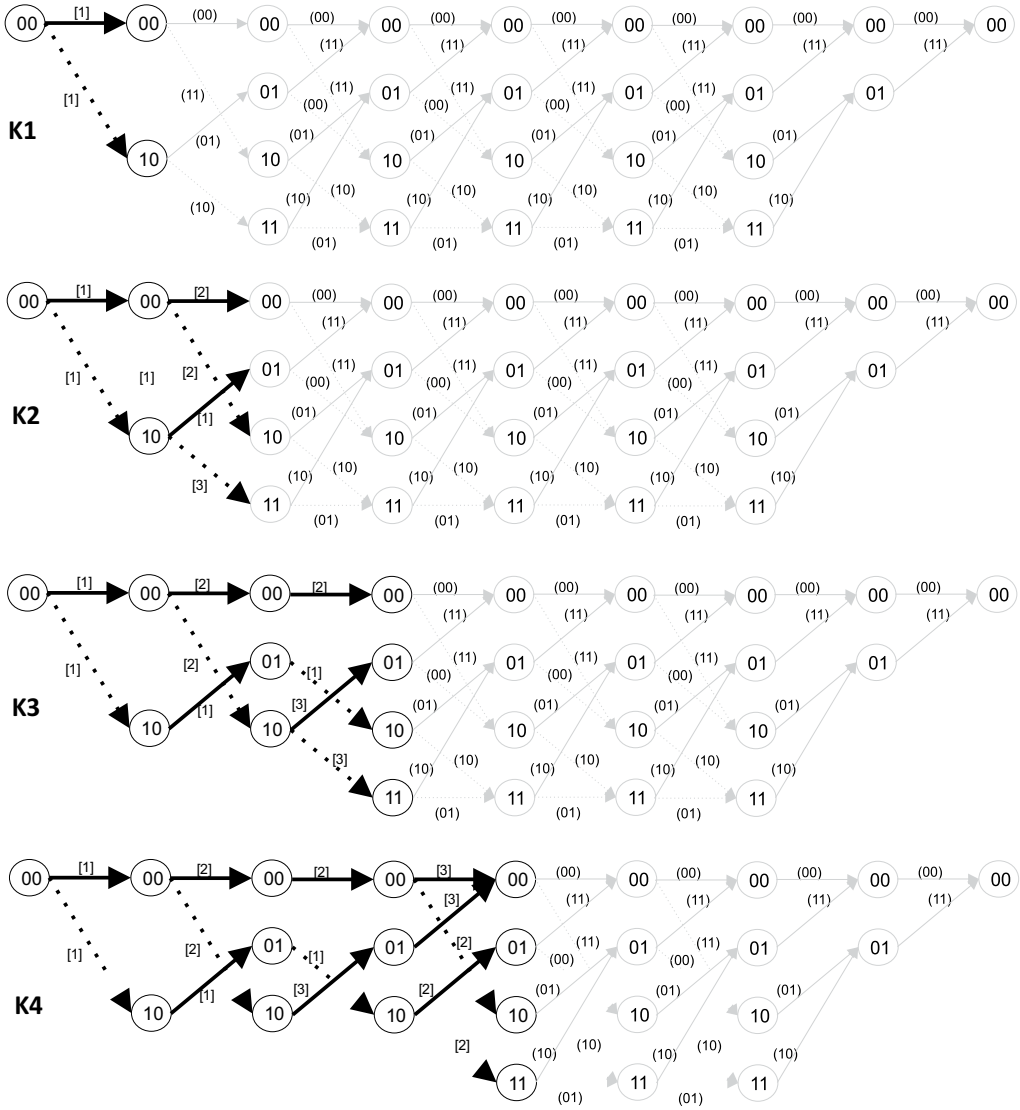
U stanje 00 težine 3 dvije putanje su moguće s ovom težinom

U stanje 01 težine 3 dvije moguće putanje

U stanje 10 težine 2 1-0-1-0-1

U stanje 11 težine 3 dvije moguće putanje.

Šesti korak je posljednji u punom trelisu. U stanje 00 možemo doći sa putanjama težine 5 (iz stanja 00) i 4 (iz stanja 01). Biramo drugu putanju. U stanje 01 možemo doći putanjama iz stanja 10 (težine 4) i stanja 11 (težine 3). Biramo drugu putanju. U stanje 10 možemo preći iz stanja 00 (težine 5) i stanja 01 (težine 4). I ovdje biramo drugu putanju. Konačno, u stanje 11 možemo preći iz stanja 10 (težine 2) i stanja 11 (težine 5). Biramo prvu putanju. Polako se kristališe optimalna putanja. Do sada su najbolji parcijalni putevi:



Slika VII.6. Putanje kroz trelis u prva četiri koraka Viterbijevog algoritma (podebljano su date najbolje parcijalne putanje u trelisu)

U stanje 00	težine 4	dvije moguće putanje
U stanje 01	težine 3	dvije moguće putanje
U stanje 10	težine 4	dvije mogućeputanje
U stanje 11	težine 2	1-0-1-0-1-1.

U sedmom koraku vršimo punjenje kodera sa 0 jer želimo da pripremimo koder za novi niz kodiranih bita. Zbog toga dolazi do smanjivanja broja parcijalnih najboljih putanja. U stanje 00 možemo doći iz 00 ili 01, a biramo drugu putanju, koja je težine 4. U stanje 01 možemo doći iz stanja 10 (težine 6) i stanja 11 (težine 2). Biramo drugu putanju. Parcijalne najbolje putanje su:

U stanje 00	težine 4	dvije moguće putanje
U stanje 10	težine 2	1-0-1-0-1-1 (0 koja nema informacioni karakter).

Konačno, u posljednjem koraku ponovo dodajemo 0, dovodeći koder u stanje 00. Prva putanja, iz stanja 00, ima težinu 6, a druga, iz stanja 01, ima težinu 2, čime ona ostaje najbolja putanja koja predstavlja dekodiranu poruku:

1 0 1 0 1 1, koja je jednaka onoj koja je poslata.

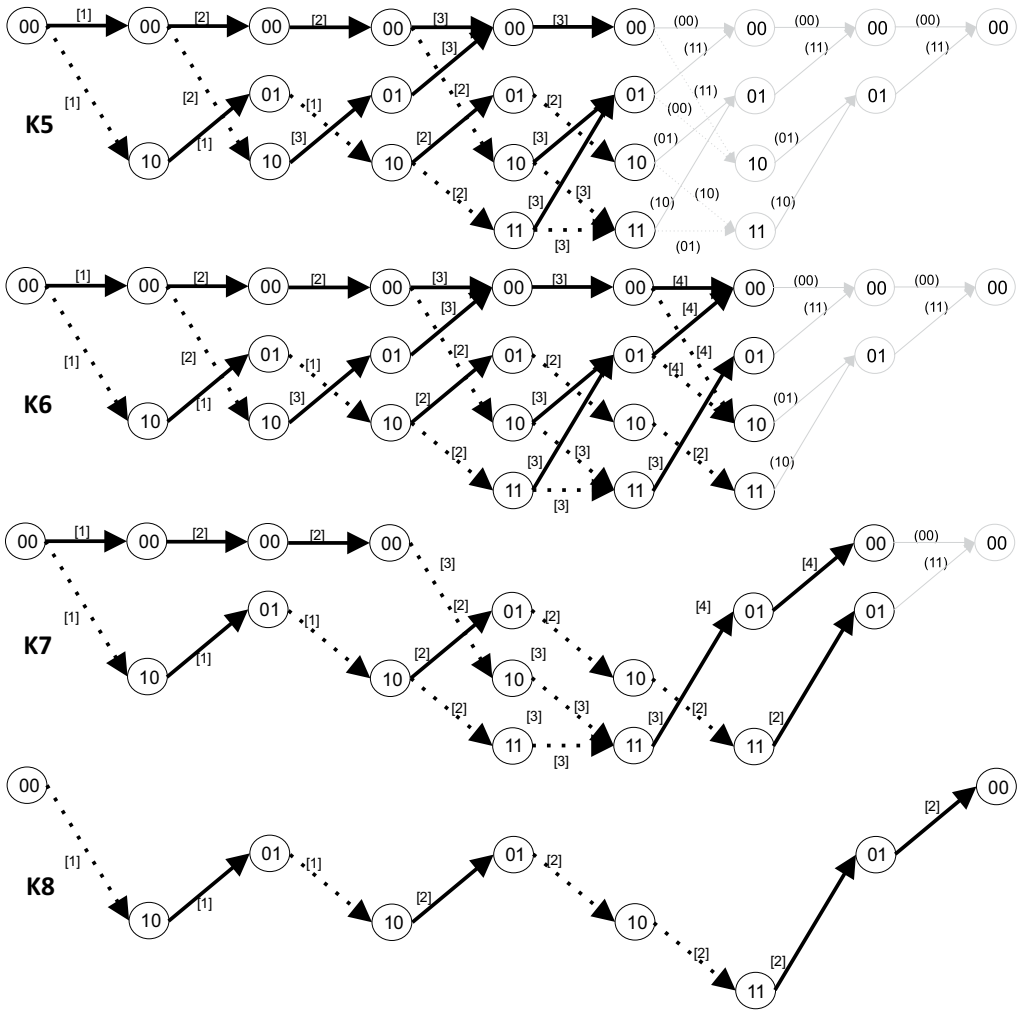
Čitav postupak je sublimiran na Slici VII.6 (za prva četiri koraka) i Slici VII.7 (za preostala četiri koraka).□

Sada možemo, suprotno uobičajenoj logici, da objasnimo tehniku sekvencijalnog dekodiranja, nakon što smo objasnili složeniji postupak Viterbijevog dekodiranja.

U prethodnom primjeru smo vidjeli da smo relativno brzo mogli da uočimo da se poslije nekoliko (četiri) koraka izdvaja putanja koja je najbolja, te da smo mogli dalje da nastavimo praćenjem te putanje. Takav način rada, kod kojeg pratimo najbolju putanju, naziva se sekvencijalnim dekodiranjem. U slučaju da nismo pošli pravim putem, doći će do akumuliranja greške. Kada Hemingovo rastojanje postane preveliko, onda znamo da nismo pošli pravim putem, pa se moramo vraćati unazad da potražimo alternativnu putanju. Naravno, nije jednostavno definisati precizan algoritam, gdje se tačno vratiti i kolika treba da bude dozvoljena akumulirana greška. Stoga, sekvencijalno dekodiranje radi dobro samo kada je procenat grešaka mali jer se u suprotnom moramo prečesto vraćati unazad u trelisu, pamtiti veliki broj putanja i kasniti s odlukom. Imajmo na umu da je konvolucioni kod zapravo i predložen za slučajeve kada su uslovi komuniciranja nepovoljni, pa je jasno da je sekvencijalni postupak male primjenljivosti.

*VII.3 Turbo-kodovi**

Podsjetimo se kodne teoreme. Kodna teorema kaže da je moguće konstruisati kod sa kodnim odnosom R, koji dekodira poruku primljenu preko kanala koji je kapaciteta C, ako je $R \leq C$, sa vjerovatnoćom greške koja je bliska nuli. Obično je i dužina kodne riječi unaprijed poznata (n), pa se ova teorema svodi na činjenicu da broj informacionih bita u riječi mora biti $k \leq nC$ da bi se mogla postići vjerovatnoća greške koja teži 0. Šenonova teorema nije konstruktivna i ne kaže kako



Slika VII.7. Putanje kroz trelis u druga četiri koraka Viterbijevog algoritma (podebljano su date najbolje parcijalne putanje u trelisu)

se do datih kodova može doći, a ujedno je asimptotska jer važi samo za $n \to \infty$. Nama je cilj da za dati kanal postignemo $k=nC$ jer alternativa podrazumijeva da šaljemo manji procenat korisnih bita, odnosno da dio energije koristimo neracionalno (svaki preneseni bit podrazumijeva potrošnju određene količine energije) da bismo nepotrebno povećali redundanciju. Još preciznije, nama je cilj da se primaknemo ovom odnosu za n konačne dužine pošto kodovi ekstremno velike dužine ili se ne mogu realizovati ili bi njihova realizacija zahtijevala neprihvatljivo velike hardverske ili softverske resurse. Drugi problem kod upotrebe predstavlja činjenica da veliko n iziskuje da imamo ekstremno veliki propusni opseg (zauzi-

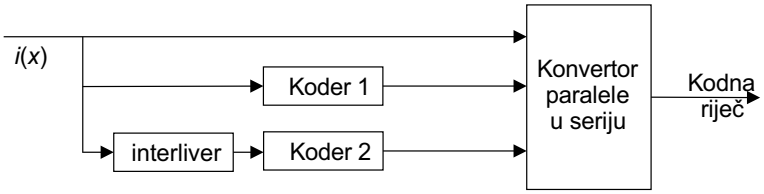
manje širokog frekvencijskog prostora, što je resurs koji je limitiran) za prenos signala kodiranog sa ovako velikim n. Zaključno sa sredinom 1990-ih, problem približavanja dostižnih granica po II Šenonovoj teoremi nije bio razriješen. I dalje su svi kodni postupci (a konvolucioni kodovi su dominirali u kombinaciji sa blok kodovima) davali rezultate s vjerovatnoćama greške koje su višestruko prevazilazile ono što je za date dužine kodne riječi bilo dostižno po II Šenonovoj teoremi i njenim ekstenzijama. Sljedstveno, bilo je potrebno uložiti veću redundanciju i trošiti više energije u prenosu informacija. Najbolji postupci su trošili po dva puta više energije nego što je prognozirano kao dostižno, po odgovarajućim teorijskim razmatranjima.

Na naučnoj konferenciji 1993. godine, grupa naučnika iz Francuske predložila je postupak „turbo-kodiranja“. Tvrdili su da se može dostići samo 12% prekomjernog trošenja energije u odnosu na prognoze II Šenonove teoreme i njenih ekstenzija. Ovakvi rezultati su u prvo vrijeme bili dočekani sa skepsom i podsmijehom, ali su ubrzo bili eksperimentalno i teorijski potvrđeni od brojnih naučnika. Nastupilo je doba turbo-kodova. Dobijeni rezultati su doveli do ekspresne primjene ovih kodnih postupaka u brojnim komunikacionim standardima. Možemo reći da je naučna zajednica dočekala ove kodove kao „žedna zemlja kišu“.

VII.3.1 Turbo-koderi

Turbo-koderi su veoma slični konvolucionim koderima. Predstavljaju kombinaciju dva ili više kodova povezanih na odgovarajući način (pogledati Sliku VII.2 za konvolucione kodove). Bitna razlika u odnosu na konvolucione kodove je postojanje interlivera. Na Slici VII.8 prikazana je jedna moguća konfiguracija turbo-kodera. U pitanju je sistematski koder sa kodnim odnosom $R = 1/3$. Kažemo da je sistematski jer se bitovi poruke direktno vode ka konvertoru serije u paralelu, te se tako pojavljuju u kodnoj riječi neizmijenjeni. Pored toga, bitovi informacione poruke do ulaza u drugi koder (Koder #2) prolaze kroz interliver i tako se permutovani kodiraju. Ovo je bitna razlika u odnosu na konvolucione kodove. Naime, sada koderi ne moraju biti različiti kao kod konvolucionih kodova, čime ne bi generisali redundanciju, već mogu biti isti jer rade na permutovanim bitima. Dok kod konvolucionog koda imamo promjenu kodera, ovdje imamo permutaciju ulazne sekvence.

Postavlja se pitanje: zbog čega je jedna ovako prosta izmjena arhitekture kodera u stanju da produkuje znatnu promjenu kvaliteta procesa kodiranja? Pokušaćemo da to pojasnimo bez posezanja za matematički rigoroznim alatima. Minimalna Hemingova distanca je bitna karakteristika nekog kodnog postupka jer utiče na karakteristike koda (mogućnost ispravljanja grešaka). Vidjeli smo da se minimalno Hemingovo rastojanje može sračunati kao najmanja težina nenulte kodne riječi.



Slika VII.8. Opšta šema sistematskog turbo-kodera sa kodnim odnosom R = 1/3

Kod konvolucionog kodiranja na raspolaganju nam je određeni broj kodera kojima treba da postignemo efekat što je moguće veće minimalne težine kodne riječi, ali očigledno da je, zbog potreba da ti koderi budu pogodne strukture i date dužine, izbor dosta skučen. Jasno je da koderi ne mogu biti isti, a ponekad se i pojedine strukture kodera moraju odbaciti. Dakle, imamo skučen skup mogućih kombinacija kodera pod datim okolnostima (npr. datom memorijom kodera). Dodatni problem je da često koderi daju sekvence sa malim brojem jedinica na izlazu kada ulazna sekvenca ima mali broj jedinica. Ovo sve govori u prilog limitiranim performansama konvolucionih kodera.

Kod turbo-kodera, pored slobode u dizajnu kodera (koja se relativno rijetko koristi), postoji sloboda u izboru interlivera. Kako sekvenca koja se permutuje interliverom može biti dugačka, stoga imamo veliku fleksibilnost u pogledu izbora koda koji daje veliko minimalno Hemingovo rastojanje. Da krajnje simplifikujemo opis. Možemo da tražimo one sekvence koje putem prvog kodera imaju malu Hemingovu težinu, a zatim da za te sekvence dizajniramo interliver tako da izlaz ima veliku Hemingovu težinu. Na taj način, kombinovani izlaz ima (relativno) veliku Hemingovu težinu, što implicira veliku minimalnu Hemingovu distancu i dobre karakteristike u pogledu broja grešaka koje se ovim kodnim postupcima mogu ispraviti.

Ne postoji obaveza da turbo-kod bude sistematski. U tom slučaju nema grane u koderu koja direktno vodi simbole prema izlazu (konvertoru paralele u seriju), već se u svim granama nalaze koderi. U pogledu tipova kodera, u pojedinim granama takođe postoji fleksibilnost, odnosno može biti bilo koji kod koji je do sada izučavan, pa i oni koji nisu izučavani. Ipak, po pravilu, to su blok ili konvolucioni koderi, realizovani putem pomjeračkog registra sa ili bez povratne sprege. Ipak, performanse kodnog postupka bitno zavise od izabranih kodera. Kodovi sa povratnom spregom obično daju bolje rezultate, u smislu vjerovatnoće greške, u procesu dekodiranja pod istim okolnostima, u odnosu na kodere koji su realizovani bez povratne sprege.

VII.3.2 Odnos maksimalne vjerodostojnosti

Očigledno, složenost turbo-kodiranja nije na strani kodera, već na strani dekodera. Ova karakteristika je zapravo zajednička za sve kodne postupke: svi oni su znatno složeniji za dekodiranje, u odnosnu na kodiranje. Bitna karakteristika koja odvaja turbo-dekodiranje od svih do sada učenih postupaka je da je ono zasnovano na **mekim ulazima**. Signal nosilac informacije je na ulazu u svaki sistem bitno izobličen u odnosu na poslati signal. Mi smo do sada uvijek pretpostavljali da je na neki način on ponovo vraćen u digitalni oblik. Obično se to postiže kolima koji integrale signal u nekom intervalu, uz odgovarajuću sinhronizaciju, a zatim se taj integraljeni signal poredi sa pragom i na osnovu toga se donosi odluka da primljeni simbol proglasimo određenim digitom – cifrom (ili bitom). Turbo-kod prima analogni signal koji može zbog izobličenja da uzme bilo koju vrijednost iz određenog intervala. Za razliku od većine drugih kodnih postupaka, umjesto da se putem komparatora ovakav primljeni signal pretvori u sekvencu nula i jedinica, turbo-dekoder radi s analognim vrijednostima. Ako je, na primjer, limiter postavljen na 0.5V i primljeni signal od 1.3V, to znači da je ovo jedinica sa mnogo većom vjerovatnoćom nego u slučaju kada je primljeni signal 0.6V, odnosno tek nešto preko napona praga od 0.5V. Radi jednostavnosti, podrazumijevaćemo da radimo sa bipolarnom sekvencom, odnosno da je bit 1 predstavljen pozitivnim naponom (strujom ili drugom veličinom), dok je bit nula prikazan signalom iste amplitude, ali drugog (negativnog) polariteta. Prirodno je pretpostaviti da je prag postavljen na nulu.

Osnovni matematički koncept koji se koristi prilikom turbo-dekodiranja je logaritamski odnos uslovnih vjerovatnoća, koji se u našoj literaturi naziva „odnosom vjerodostojnosti“ (engl. *likelihood ratio* – LR), ili odgovarajući logaritamski odnos − LLR. Ovaj odnos ima ogromnu primjenu u nauci i jedan je od osnova testiranja hipoteza, teorije estimacije (procjene) itd. Testiranje hipoteza koje mi provodimo na osnovu primljenog signala, kada testiramo koji je ulazni signal mogao biti poslat, zasnovano je na konceptu uslovnih vjerovatnoća i Bajesovom pravilu. Podsjetimo se ukratko nekih odnosnih koncepata. Veza između uslovne, združene i marginalne vjerovatnoće je:

$$P(B|A)P(A) = P(A|B)P(B) = P(AB),$$

iz čega slijedi:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}.$$

Predmetna relacija uspostavlja vezu uslovnih vjerovatnoća i naziva se Bajesovom. U slučaju kada baratamo porukama koje su poslate iz nekog diskretnog alfabeta, a

primamo vrijednosti koje su iz kontinualnog domena, prethodnu relaciju možemo napisati na sljedeći način:

$$P(d=i\,|\,x)=\frac{p(x\,|\,d=i)P(d=i)}{p(x)},$$

gdje je:

$$p(x)=\sum_{i=1}^{M}p(x\,|\,d=i)P(d=i).$$

Pretpostavljamo da je $d=i$ neki mogući događaj iz diskretnog skupa događaja – alfabeta (recimo unipolarnog $\{0, 1\}$ ili bipolarnog binarnog $\{-1, 1\}$), dok su $p(x)$ i $p(x|d=i)$ funkcije rasporedjele kontinualne slučajne promjenljive x. Iz prethodne relacije je očigledno da je $p(x)$ nezavisno od i. Pretpostavimo da imamo binarni alfabet, a da su vrijednosti iz ovog alfabeta prikazane fizičkim vrijednostima $+1$ i -1 (recimo u voltima, amperima itd.). Dakle, pretpostavljamo da d može uzeti te dvije vrijednosti. Za kanal u kome slučajni proces može biti modelovan kao Gausov bijeli šum, uslovne vjerovatnoće $p(x|d=1)$ i $p(x|d=-1)$ prikazane su na Slici VII.9. Predmetne uslovne gustine raspodjele nazivaju se funkcijama vjerodostojnosti. Pretpostavimo da imamo primljenu vrijednost x_k, označenu na slici. U predmetnom slučaju mnogo je vjerovatnije da ova vrijednost odgovara situaciji da je primljeno $d=1$, a ne $d=-1$; međutim, ni ta druga opcija nije isključena. Dakle, odluka se ne može donijeti sa sigurnošću, već sa određenom vjerovatnoćom koja zavisi od odnosa vjerodostojnosti, odnosno odnosa uslovnih vjerovatnoća. Ovakvo odlučivanje se naziva „mekim" (engl. *soft*), dok se odlučivanje kod kojeg donosimo odluku samo poređenjem sa pragom naziva „tvrdim" (engl. *hard*). Tvrda odluka se može zapisati na sljedeći način:

$$\begin{cases} p(d=1\,|\,x) > p(d=-1\,|\,x) & \text{dekodiramo } 1 \\ p(d=1\,|\,x) < p(d=-1\,|\,x) & \text{dekodiramo } -1 \text{ (odnosno } 0). \end{cases}$$

Putem Bajesove formule generalnije možemo izraziti ovaj odnos na sljedeći način. Ako je:

$$p(x\,|\,d=1)P(d=1) > p(x\,|\,d=-1)P(d=-1),$$

primljen je simbol 1, a u suprotnosti, primljen je simbol 0. Ovo se češće izražava preko korespondentnog odnosa vjerovatnoća (odnosa vjerodostojnosti):

$$\frac{p(x\,|\,d=1)}{p(x\,|\,d=-1)}\frac{P(d=1)}{P(d=-1)},$$

te se tvrda odluka donosi poređenjem ovog odnosa sa jedinicom. Kako ove vrijednosti mogu biti jako male ili veoma velike, umjesto ovakvog odnosa, a i iz drugih razloga, koristi se logaritamski odnos

$$L(d\,|\,x)=\log\left[\frac{P(d=1\,|\,x)}{P(d=-1\,|\,x)}\right]=\log\left[\frac{p(x\,|\,d=1)}{p(x\,|\,d=-1)}\frac{P(d=1)}{P(d=-1)}\right].$$

Algebra u domenu vjerodostojnosti je dosta interesantna i od posebnog je značaja za dekodiranje kodova. Iz logaritamskog odnosa:

$$L(d)=\log\frac{P(d=1)}{P(d=-1)}=\log\frac{P(d=1)}{1-P(d=1)},$$

moguće je odrediti vjerovatnoće pojedinih događaja:

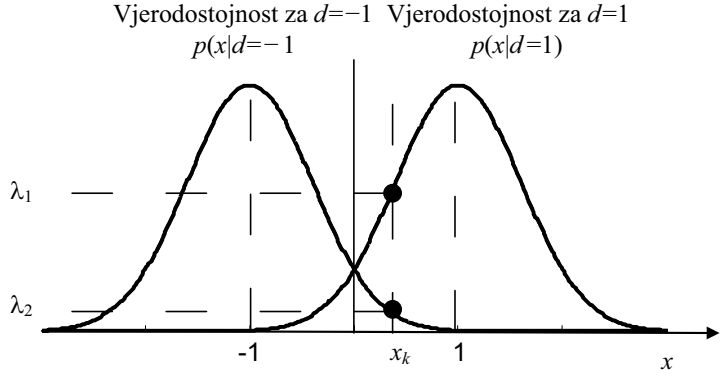
$$P(d=1)=\frac{e^{L(d)}}{1+e^{L(d)}}\qquad\text{dok je}\qquad P(d=-1)=\frac{1}{1+e^{L(d)}},$$

gdje je uzeta osnova logaritma e. Slično važi i za uslovne vjerovatnoće i njihovo određivanje iz odnosa $L(d|x)$.

Bitna veličina koja se pojavljuje u razvoju procesa dekodiranja turbo-kodova je odnos vjerodostojnosti zbira dva simbola d_1+d_2, odnosno $d_1\oplus d_2$:

$$L(d_1\oplus d_2)=\log\frac{P(d_1\oplus d_2=1)}{P(d_1\oplus d_2=-1)}=\log\frac{P(d_1=1)P(d_2=-1)+P(d_1=-1)P(d_2=1)}{P(d_1=1)P(d_2=1)+P(d_1=-1)P(d_2=-1)}$$

$$=\log\frac{\frac{e^{L(d_1)}}{1+e^{L(d_1)}}\frac{1}{1+e^{L(d_2)}}+\frac{1}{1+e^{L(d_1)}}\frac{e^{L(d_2)}}{1+e^{L(d_2)}}}{\frac{e^{L(d_1)}}{1+e^{L(d_1)}}\frac{e^{L(d_2)}}{1+e^{L(d_2)}}+\frac{1}{1+e^{L(d_1)}}\frac{1}{1+e^{L(d_2)}}}=\log\frac{e^{L(d_1)}+e^{L(d_2)}}{e^{L(d_1)+L(d_2)}+1}.$$



Slika VII.9. Uslovne vjerovatnoće sa označenim vjerovatnoćama da vrijednosti x_k odgovara poslati simbol $d=1$, odnosno $d=-1$

Premda izgleda jednostavno, ovakva formula može biti problematična. U realnim sistemima barata se ogromnim brojem podataka – bita, pa je računanje logaritamske ili eksponencijalne funkcije nepraktično. Stoga se često upotrebljavaju aproksimacije kao što je:

$$L(d_1 \oplus d_2) \approx (-1)\text{sign}[L(d_1)]\text{sign}[L(d_2)]\min[|L(d_1)|,|L(d_2)|],$$

gdje sign predstavlja funkciju znaka, dok je min minimum argumenata. Uzmimo, recimo, da je $L(d_1)=1$, a $L(d_2)=0.5$. Tačan izraz daje –0.2273, dok približan daje –0.5. Situacija je znatno bolja ako je razlika odnosa vjerodostojnosti veća, tako da za $L(d_1)=4$ i $L(d_2)=1$ tačan izraz daje –0.9843, dok približan daje –1. Dakle, spremni smo da zbog jednostavnijeg računanja platimo određenu nepreciznost u računanju logaritamskih odnosa.

VII.3.3 Dekoder sa mekim ulazom i mekim izlazom

Srce sistema za dekodiranje kod turbo-kodova je dekoder sa mekim ulazom i mekim izlazom (engl. *soft-input soft-output* − SISO). Osnovna namjena ovog dekodera, prikazanog na Slici VII.10, jeste proračun logaritamskog odnosa vjerovatnoća za primljenu sekvencu. Sistem je baziran na iterativnom postupku. Zapišimo $L(d|x)$ kao:

$$L(d \mid x) = \log\left[\frac{p(x \mid d=1)}{p(x \mid d=-1)}\right] + \log\left[\frac{P(d=1)}{P(d=-1)}\right] = L(x \mid d) + L(d),$$

gdje je $L(x|d)$ logaritamski odnos izlaza iz kanala x, pod uslovom da je primljeno $d=1$ ili $d=-1$, dok je $L(d)$ logaritamski odnos vjerovatnoća simbola koje generiše predajnik, što je apriori (unaprijed) poznato. Ovo se može pojednostavljeno zapisati kao:

$$L'(\hat{d}) = L_C(x) + L(d),$$

gdje $L_C(x)$ označava LLR koji je rezultat mjerenja napravljenog na detektoru, a $L'(\hat{d})$ LLR za bitove van detektora (ulaz u dekoder). Sve prethodne jednačine su uvedene imajući na umu samo dekoder podataka. Međutim, uvodeći informaciju

Slika VII.10. Šematski prikaz određivanja logaritamskog odnosa vjerovatnoća potrebnog za donošenje odluke kod SISO dekodera

o procesu kodiranja, imaćemo neke dobitke i u procesu donošenja odluka. Za sistematske kodove LLR (meki izlaz) $L(\hat{d})$ je jednak:

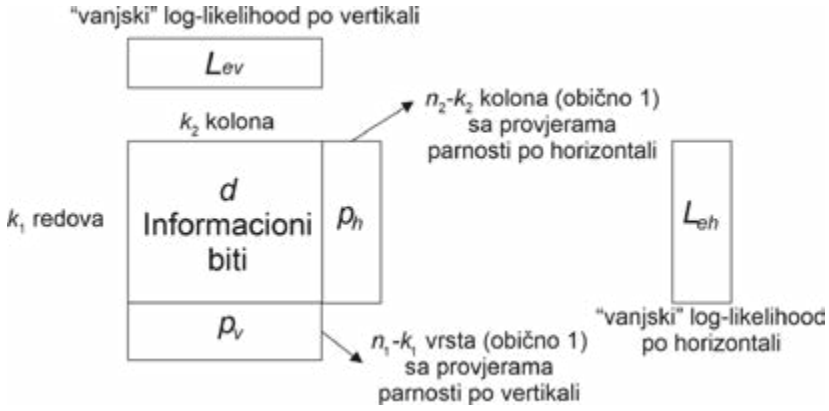
$$L(\hat{d}) = L'(\hat{d}) + L_e(\hat{d}),$$

gdje je $L_e(\hat{d})$ LLR dobijen dodatnim znanjem o procesu dekodiranja. Izlazna sekvenca sistematskog dekodera sastoji se od bitova poruke i bitova parnosti. Stoga se prethodni odnos vjerodostojnosti može shvatiti kao suma LLR-ova za poruku i dodatak koji se može dobiti na osnovu bitova parnosti. Konačno možemo zapisati:

$$L(\hat{d}) = L_C(x) + L(d) + L_e(\hat{d}).$$

Meka odluka $L(\hat{d})$ je realni (meki) broj na osnovu kojeg donosimo odluku (tvrdu) sa određenom sigurnošću. Odluka se donosi na osnovu znaka $L(\hat{d})$, odnosno za pozitivne vrijednosti odlučujemo da je $d = 1$, dok u suprotnom donosimo zaključak da je $d = -1$. Apsolutna vrijednost $L(\hat{d})$ ukazuje na sigurnost sa kojom se odluka donosi. U prvoj iteraciji u dekodiranju, SISO dekoder obično pretpostavlja da su svi binarni podaci jednako vjerovatni, dajući inicijalnu apriori LLR vrijednost $L(d) = 0$ za treći član u prethodnoj jednakosti. Predetekciona vrijednost kanala $L_C(x)$ određena je formirajući logaritam odnosa λ_1 i λ_2. Vrijednost $L(\hat{d})$ dobijena je na osnovu logaritamskog odnosa za dekoder $L'(\hat{d})$ i spoljnjeg odnosa $L_e(\hat{d})$, koji predstavlja znanje do kojeg se došlo u procesu dekodiranja. Tokom procesa iterativnog dekodiranja, spoljni LLR se povratnom spregom ponovo dovodi na ulaz dekodera u cilju poboljšavanja apriori vrijednosti u narednoj iteraciji.

Radi ilustracije funkcionisanja procesa kod SISO dekodera, posmatrajmo pravougaoni kod prikazan na Slici VII.11. Pravougaoni kod se sastoji od informacionih



Slika VII.11. Ilustrativna šema pravougaonog koda sa korespondentnim logaritamskim odnosima vjerodostojnosti

bita smještenih u matricu, dimenzija $k_1 \times k_2$, kojima su dodati biti parnosti kolona (sa bitima parnosti po vrstama). Elementi strukture su označeni na sljedeći način: d – informacioni biti, p_h horizontalni bitovi parnosti i p_v vertikalni bitovi parnosti. Blokovi označeni sa L_{eh} i L_{ev} daju vanjske logaritamske odnose vjerodostojnosti, određene za horizontalni i vertikalni dekodirajući korak, respektivno. Napomenimo da je pravougaoni kod jednostavan primjer koji nije od naročite praktične važnosti, ali da može da posluži da na računskom primjeru objasnimo proces dekodiranja. Njegova struktura se sastoji od dva odvojena dekodirajuća koraka – horizontalnog i vertikalnog. Algoritam za iterativno dekodiranje za pravougaoni kod sastoji se od sljedećih koraka:

1. Postaviti apriori informacije $L(d) = 0$ (suštinski, pretpostavljena je ista vjerovatnoća nula i jedinica u kodnoj riječi).
2. Dekodirati po horizontali i odrediti horizontalnu spoljnu informaciju:

$$L_{eh}(\hat{d}) = L(\hat{d}) - L_C(x) - L(d).$$

3. Postavi se da je $L(d) = L_{eh}(\hat{d})$.
4. Dekodirati po vertikali i dobiti:

$$L_{ev}(\hat{d}) = L(\hat{d}) - L_C(x) - L(d).$$

5. Postavi se da je $L(d) = L_{ev}(\hat{d})$.
6. Ako se nakon iteracije može donijeti validna odluka, preći na korak 7, a u suprotnom ponoviti korak 2.
7. Meki izlaz se računa kao:

$$L(\hat{d}) = L_C(x) + L_{eh}(\hat{d}) + L_{ev}(\hat{d}).$$

Označimo informacione bite kao d_i, $i \in [1, k_1 \times k_2]$, a biti parnosti koji provjeravaju informacione bite neka su označeni kao $p_{i_1 i_2 \cdots i_k}$, gdje vrijednosti u indeksu p predstavljaju informacione bite koji učestvuju u predmetnoj provjeri, bilo po vertikali ili horizontali. Radi jednostavnosti posmatraćemo kod sa 2×2 informacionih bita, označenih d_1, d_2, d_3 i d_4. Posmatraćemo samo 4 provjere parnosti (bez provjere parnosti po čitavoj matrici). Tako je dobijeni pravougaoni kod dimenzija (8,4). Provjere parnosti po horizontali označene su kao p_{12} i p_{34}, dok su po vertikali p_{13} i p_{24}. Kodna riječ se može zapisati kao sekvenca: $d_1 d_2 d_3 d_4 p_{12} p_{34} p_{13} p_{24}$. U predmetnom slučaju između bitova parnosti i informacionih bita važe sljedeće relacije:

$$d_i \oplus d_j = p_{ij} \qquad d_i = d_j \oplus p_{ij} \qquad d_j = d_i \oplus p_{ij}.$$

Na prijemu, umjesto ove sekvence, primamo „meku“ sekvencu $[x_i, x_{ij}]$, gdje x_i predstavljaju odgovarajuće vrijednosti na pozicijama informacionih bita, dok su x_{ij} odgovarajuće vrijednosti na pozicijama bita parnosti. Smatramo da su svi primljeni biti bili pod uticajem šuma. Radi jednostavnosti, uzmimo da je u pitanju Gausov šum: $x_i = d_i + \nu_i$, odnosno $x_{ij} = p_{ij} + \nu_{ij}$. Pretpostavićemo da je šum i.i.d. Uvešćemo oznaku da je vrijednost signala na poziciji svakog primljenog bita označena kao $\{x_k\}$, gdje se k može tretirati kao vremenski indeks. Koristivši prethodne relacije i pretpostavljajući da je aditivni šum gausovski, možemo odgovarajući logaritamski odnos vjerodostojnosti zapisati kao:

$$L_C(x_k) = \log\left[\frac{p(x_k \mid d=1)}{p(x_k \mid d=-1)}\right] = \log\left[\frac{\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\frac{(x_k-1)^2}{\sigma^2}\right]}{\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\frac{(x_k+1)^2}{\sigma^2}\right]}\right] =$$

$$= -\frac{1}{2}\left(\frac{x_k-1}{\sigma}\right)^2 + \frac{1}{2}\left(\frac{x_k+1}{\sigma}\right)^2 = \frac{2}{\sigma^2}x_k.$$

Usvojimo da je varijansa $\sigma^2 = 1$, pa dobijamo:

$$L_C(x_k) = 2x_k.$$

Posmatrajmo sljedeći primjer, gdje je sekvenca podataka $d_1d_2d_3d_4$ kreirana od binarnih cifara 1001, kao što je prikazano u Tabeli VII.1. Na osnovu uvedenih relacija slijedi da su biti parnosti: $p_{12}p_{34}p_{13}p_{24}$ jednaki 1111. Dakle, poslata sekvenca je jednaka:

$$[d_i, p_{ij}] = 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1.$$

Neka je poruka prenesena bipolarnim impulsima $+1\ -1\ -1\ +1\ +1\ +1\ +1\ +1$ i neka zbog šuma u prenosu dobijena poruka ima oblik:

$$[x_k] = 0.75\ 0.05\ 0.10\ 0.15\ 1.25\ 1.00\ 3.00\ 0.50.$$

Logaritamski odnos vjerodostojnosti dobijen iz kanala u ovom slučaju je (uz prethodno usvojeno pojednostavljenje radi se o dvostrukoj vrijednosti primljenih simbola iz kanala):

$$[L_C(x_k)] = 1.50\ 0.10\ 0.20\ 0.30\ 2.50\ 2.00\ 6.00\ 1.00.$$

Radi pregledosti, dali smo ove odnose u Tabeli VII.2.

Zbog toga što je na drugom i trećem bitu logaritamski odnos vjerodostojnosti pozitivan, kod bi iskazivao greške na ta dva bita. Prije nego pokažemo da ovakva greška nije problematična u procesu dekodiranja turbo-kodova, moramo se ponovo

$d_1 = 1$	$d_2 = 0$	$p_{12} = 1$
$d_3 = 0$	$d_4 = 1$	$p_{34} = 1$
$p_{13} = 1$	$p_{24} = 1$	

Tabela VII.1. Ispravna poruka poslata u kanal za pravougaoni kod (8,4) prikazana tabelarno

$L_C(x_1) = 1.5$	$L_C(x_2) = 0.1$	$L_C(x_{12}) = 2.5$
$L_C(x_3) = 0.2$	$L_C(x_4) = 0.3$	$L_C(x_{34}) = 2.0$
$L_C(x_{13}) = 6.0$	$L_C(x_{24}) = 1.0$	

Tabela VII.2. Logaritamski odnosi vjerodostojnosti dobijeni iz kanala za pravougaoni kod (8,4)

osvrnuti na algebru koja se koristi za računanje logaritamskih odnosa vjerodostojnosti. Podsjetimo se ranije uvedene aproksimacije:

$$L(d_1)\Diamond(d_2) = L(d_1 \oplus d_2) \approx (-1)\text{sign}[L(d_1)]\text{sign}[L(d_2)]\min[|L(d_1)|,|L(d_2)|],$$

gdje smo radi pojednostavljenja uveli novi operator $\Diamond$. Za uvedeni pravougaoni kod koristimo relaciju da se meki izlaz $L(\hat{d}_1)$ za d_1 računa kao:

$$L(\hat{d}_1) = L_C(x_1) + L(d_1) + ([L_C(x_2) + L(d_2)]\Diamond L_C(x_{12})),$$

gdje je $([L_C(x_2) + L(d_2)]\Diamond L_C(x_{12}))$ spoljna vrijednost koja pomaže kodu (signal koji odgovara podacima d_2 i njegovoj apriori vjerovatnoći i signal koji predstavlja bite parnosti). U opštem slučaju, meki izlaz $L(\hat{d}_i)$ za primljeni signal d_i koji odgovara podacima je:

$$L(\hat{d}_i) = L_C(x_i) + L(d_i) + ([L_C(x_j) + L(d_j)]\Diamond L_C(x_{ij})),$$

gdje su $L_C(x_i)$, $L_C(x_j)$ i $L_C(x_{ij})$ logaritamski odnosi vjerodostojnosti dobijeni iz kanala na pozicijama koje korespondiraju odgovarajućim simbolma d_i, d_j i p_{ij}. $L(d_i)$ i $L(d_j)$ su LLR apriornih vjerovatnoća za d_i i d_j respektivno, dok je $([L_c(x_j) + L(d_j)]\Diamond L_c(x_{ij}))$ spoljni doprinos koda. Na primjer, sada je meki izlaz za bit d_1 $L(\hat{d}_1)$ unaprijeđen na osnovu LLR odnosa za podatak d_2 i za parnost p_{12}. Horizontalne $L_{eh}(\hat{d})$ i vertikalne kalkulacije $L_{ev}(\hat{d})$ izražavaju se na sljedeći način:

$$L_{eh}(\hat{d}_1) = [L_C(x_2) + L(d_2)]\Diamond L_c(x_{12})$$
$$L_{ev}(\hat{d}_1) = [L_C(x_3) + L(d_3)]\Diamond L_C(x_{13})$$
$$L_{eh}(\hat{d}_2) = [L_C(x_1) + L(d_1)]\Diamond L_c(x_{12})$$

$$L_{ev}(\hat{d}_2) = [L_C(x_4) + L(d_4)] \lozenge L_C(x_{23})$$
$$L_{eh}(\hat{d}_3) = [L_C(x_4) + L(d_4)] \lozenge L_C(x_{34})$$
$$L_{ev}(\hat{d}_3) = [L_C(x_1) + L(d_1)] \lozenge L_C(x_{13})$$
$$L_{eh}(\hat{d}_4) = [L_C(x_3) + L(d_3)] \lozenge L_C(x_{34})$$
$$L_{ev}(\hat{d}_4) = [L_C(x_2) + L(d_2)] \lozenge L_C(x_{24}).$$

Kada uvrstimo LLR vrijednosti iz tabele u izraze za $L_{eh}(\hat{d})$, uz inicijalno postavljanje $L(d)$ vrijednosti na nulu, dobijamo:

$$L_{eh}(\hat{d}_1) = (0.1 + 0) \lozenge 2.5 = -0.1 = \text{novi } L(d_1)$$
$$L_{eh}(\hat{d}_2) = (1.5 + 0) \lozenge 2.5 = -1.5 = \text{novi } L(d_2)$$
$$L_{eh}(\hat{d}_3) = (0.3 + 0) \lozenge 2.0 = -0.3 = \text{novi } L(d_3)$$
$$L_{eh}(\hat{d}_4) = (0.2 + 0) \lozenge 2.0 = -0.2 = \text{novi } L(d_4),$$

gdje je LLR sabiranje vršeno upotrebom uvedene aproksimacije. Sada obavljamo računanje prve vertikalne LLR vrijednosti $L_{ev}(\hat{d})$. Za $L(d)$ koristimo vrijednosti koje su sračunate u prethodnom koraku za $L_{eh}(\hat{d})$:

$$L_{ev}(\hat{d}_1) = (0.2 - 0.3) \lozenge 6.0 = 0.1 = \text{novi } L(d_1)$$
$$L_{ev}(\hat{d}_2) = (0.3 - 0.2) \lozenge 1.0 = -0.1 = \text{novi } L(d_2)$$
$$L_{ev}(\hat{d}_3) = (1.5 - 0.1) \lozenge 6.0 = -1.4 = \text{novi } L(d_3)$$
$$L_{ev}(\hat{d}_4) = (0.1 - 1.5) \lozenge 1.0 = 1.0 = \text{novi } L(d_4).$$

Rezultati prve pune iteracije za dva dekodirajuća koraka (horizontalni i vertikalni) prikazani su u Tabeli VII.3. Svaki dekodirajući korak popravlja originalni LLR. Ovo se može vidjeti na osnovu računanja LLR pomoću uvedene aproksimacije. Popravka koja je ostvarena kada se na polazni LLR provedu horizontalni i spoljni LLR data je u Tabeli VII.4. Originalni LLR, sa dodatim horizontalnim i vertikal-

1.5	0.1
0.2	0.3

$L_c(x_k)$

−0.1	−1.5
−0.3	−0.2

$L_{eh}(\hat{d})$ nakon prvog horizontalnog koraka

0.1	−0.1
−1.4	1.0

$L_{ev}(\hat{d})$ nakon prvog vertikalnog koraka

Tabela VII.3. LLR-ovi $L_c(x_k)$, $L_{eh}(\hat{d})$ i $L_{ev}(\hat{d})$

1.4	− 1.4
− 0.1	0.1

Tabela VII.4. Popravka LLR uzrokovana $L_{eh}(\hat{d})$

1.5	− 1.5
− 1.5	1.1

Tabela VII.5. Popravka LLR postignuta sa $L_{eh}(\hat{d}) + L_{ev}(\hat{d})$

1.5	0.1
0.2	0.3

$L_c(x_k)$

0	− 1.6
− 1.3	− 1.2

$L_{eh}(\hat{d})$

nakon drugog horizontalnog koraka

1.1	− 1.0
− 1.5	1.0

$L_{ev}(\hat{d})$

nakon drugog vertikalnog koraka

Tabela VII.6. LLR-ovi $L_c(x_k)$, $L_{eh}(\hat{d})$ i $L_{ev}(\hat{d})$ nakon drugog koraka

2.6	− 2.5
− 2.6	2.5

Tabela VII.7. Dobijeni meki izlazi

nim spoljnim vrijednostima, daje popravku iz Tabele VII.5. Iz primjera možemo da vidimo da isključivo pomoću horizontalnih bitova parnosti dobijamo korektnu (*hard*) odluku o izlazu iz dekodera, ali sa malom sigurnošću za bite d_3 i d_4.

Nakon što dekoderom popravimo LLR sa vertikalnim spoljnim odnosom, novi LLR daje veću sigurnost. Posmatrajmo jednu dodatnu horizontalnu i vertikalnu dekodirajuću iteraciju da vidimo da li postoji neko dodatno značajno poboljšanje rezultata. Ponavljamo praktično istu proceduru koja je prethodno odrađena:

$$L_{eh}(\hat{d}_1) = (0.1 - 0.1)\Diamond 2.5 = 0 = \text{nova } L(d_1)$$

$$L_{eh}(\hat{d}_2) = (1.5 + 0.1)\Diamond 2.5 = -1.6 = \text{nova } L(d_2)$$

$$L_{eh}(\hat{d}_3) = (0.3 + 0.1)\Diamond 2.0 = -1.3 = \text{nova } L(d_3)$$

$$L_{eh}(\hat{d}_4) = (0.2 - 1.4)\Diamond 2.0 = 1.2 = \text{nova } L(d_4).$$

U sljedećem koraku nastavljamo s drugom vertikalnom kalkulacijom da bismo izračunali $L_{ev}(\hat{d})$, koristivši novu $L(d)$ iz druge horizontalne relacije:

$$L_{ev}(\hat{d}_1) = (0.2 - 1.3)\Diamond 6.0 = 1.1 = \text{nova } L(d_1)$$

$$L_{ev}(\hat{d}_2) = (0.3 + 1.2) \lozenge 1.0 = -1.0 = \text{nova } L(d_2)$$
$$L_{ev}(\hat{d}_3) = (1.5 + 0) \lozenge 6.0 = -1.5 = \text{nova } L(d_3)$$
$$L_{ev}(\hat{d}_4) = (0.1 - 1.6) \lozenge 1.0 = 1.0 = \text{nova } L(d_4).$$

Nakon druge iteracije u dekodiranju, gdje smo ažurirali logaritamske odnose vjerodostojnosti po horizontalama i vertikalama, meki izlaz se ponovo računa kao:

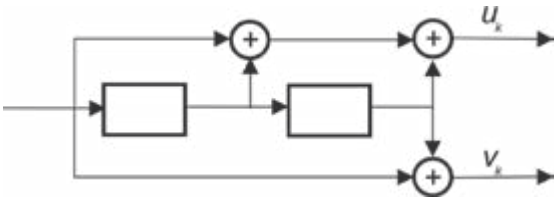
$$L(\hat{d}) = L_c(x) + L_{eh}(\hat{d}) + L_{ev}(\hat{d}).$$

Finalna horizontalna i vertikalna spoljna vrijednost i rezultujuća dekodirajuća LLR vrijednost prikazane su u Tabeli VII.6 i Tabeli VII.7. U ovom primjeru druga kompletna iteracija za oba koda (ukupno 4 iteracije) sugeriše umjereno poboljšanje u odnosu na samo jednu iteraciju.

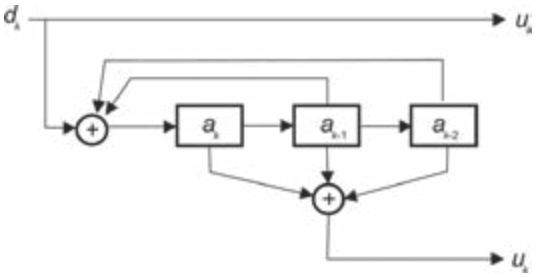
VII.3.4. Turbo-kodovi korak bliže stvarnoj realizaciji

Nakon što smo opisali osnovne koncepte nadovezivanja, iteracija i mekog odlučivanja korišćenjem pravougaonog koda, sada ćemo ove ideje primijeniti na implementaciju turbo-kodova. Date su samo grube informacije o ovom procesu, bez namjere da se detaljno izučava. Turbo-kod ćemo formirati kao paralelno nadovezivanje komponenti kodova. Kao kod jednostavnog binarnog konvolucionog kodera imamo $R = 1/2$ sa ograničenom dužinom K i memorijom $K - 1$. Ulaz u koder u trenutku k je bit d_k i odgovarajuća kodna riječ je par bitova (u_k, v_k), gdje su koeficijenti pojedinih generatorskih polinoma dati kao $\mathbf{G}_1 = [g_{1i}]$ i $\mathbf{G}_2 = [g_{2i}]$. Jedan mogući ovakav koder dat je na Slici VII.12. Napominjemo da je dati koder nesistematski i da ima konačni impulsni odziv (jedan ulazni bit ima uticaj na konačan broj izlaznih bita, u ovom slučaju 3). Ograničena dužina je $K = 3$ i dva generatora koda su opisana sa $\mathbf{G}_1 = [111]$ i $\mathbf{G}_2 = [101]$. Napominjemo da metoda kodiranja (nesistematski ili sistematski sa beskonačnim odzivom, odnosno za rekurzivne sistematske kodove) ne utiče toliko na proces dekodiranja, koliko na performanse dekodiranja za signale zahvaćene šumom. Pokazuje se da za veliki odnos signal−šum (mali uticaj šuma) nesistematski kodovi rade znatno bolje, dok za mali odnos signal−šum (veliki uticaj šuma) bolje rade sistematski rekurzivni koderi. Kako se turbo-kodovi ipak dizajniraju da rade za složenije uslove, onda se češće primjenjuju sistematski kodovi, s tim što neki sistemi posjeduju mogućnost da vrše „prebacivanje“ sa jednog na drugi kodirajući metod u zavisnosti od stanja u kanalu. Primjer sistematskog rekurzivnog kodera koji se koristi kod turbo-kodiranja, kao i način na koji se dva ovakva kodera mogu povezati u turbo-koder prikazani su na Slici VII.13.

Već smo ukazali na razloge zbog kojih je važan interliver i zbog čega ovakav kodni sistem dobro radi kod rekurzivnih kodova. Ipak, zbog značaja, vrijedi ovo



Slika VII.12. Konvolucioni koder iz Poglavlja VII.1



Slika VII.13. Sistematski rekurzivni konvolucioni koder

još jednom apostrofirati. Da bi neki kodni sistem imao dobre performanse u otklanjanju grešaka, mora da ima što veću minimalnu Hemingovu distancu, odnosno što veću moguću minimalnu Hemingovu težinu nenultih kodnih riječi. Odnosno, cilj je da broj jedinica u riječi koja ih ima najmanje bude što je moguće veći. Kod našeg sistema u pojedinačnim koderima situacija da se na izlazu pojavi riječ sa malim brojem jedinica ne može se izbjeći, ali se može izbjeći da istovremeno oba kodera produkuju riječ sa malim brojem jedinica. Uparivanje riječi sa malim brojem jedinica izbjegava se pažljivim dizajnom interlivera.

Interliver može da ima dobru kontrolu nad težinom kodne riječi samo ako je kod rekurzivan. Kod rekurzivnih kodova ono što je na prvi pogled najgora ulazna sekvenca (sekvenca sa jednom jedinicom i ostalim nulama) nije kritično jer rekurzivni kodovi rade na principu filtera sa beskonačnim impulsnim odzivom, odnosno jedna jedinica će produkovati beskonačno mnogo jedinica na izlazu. Suprotno logici, povećanje broja jedinica u ulaznoj sekvenci ne mora da produkuje izlazne riječi sa većim brojem jedinica. Naime, može da se dogodi ono što se kod teorije filtera sa beskonačnim impulsnim odzivom naziva „poništavanjem para nula-pol“ i da za neke ulazne sekvence dobijemo konačan odziv. Za određenu ulaznu sekvencu može se dogoditi da na izlazu dobijemo relativno mali broj jedinica. Kod predmetnog sistema jedna ulazna jedinica daje na izlazu beskonačan odziv. Kritične sekvence su sa dvije jedinice koje su razmaknute sa dvije nule (svi ostali bitovi su nule) i sa tri uzastopne jedinice i ostalim nulama. Dakle, interliver u predmetnom slučaju mora biti dizajniran tako da za date slučajeve ne produkuje u drugoj grani ulaznu sekvencu koja ima iste osobine kao u prvoj grani. Na Slici VII.14 prikazan

je turbo-koder sa dva koda razdvojena interliverom. U granama se nalaze dva rekurzivna konvoluciona kodera, a ulaz se vodi direktno na izlaz u trećoj grani, pa je koder sistematski. Uočljivo je, naravno, da će samo prvi koder, u ukupnom dizajnu turbo-kodera, biti rekurzivan jer je kod drugog kodera direktna grana dobijena nakon permutovanja, odnosno premetanja originalne sekvence putem interlivera. Sa stanovišta dekodiranja ova permutacija nije problematična i može se uvijek ispravno dekodirati. Prethodno smo se na primjeru jednog jednostavnog pravougaonog koda upoznali s osnovnim principima turbo-dekodiranja. Nadamo se da je ta ideja jasna. Nažalost, fundament je jedno, a stvarna realizacija ipak je znatno složenija.

U praktičnu realizaciju ovog sistema zapravo nećemo ni ulaziti, iz razloga što za predmetne konvolucione kodove koji su dio sistema turbo-kodiranja taj dekodirajući stepen sadrži komplikovane tehnike za dekodiranje, koje su naslijeđene iz konvolucionih kodova, a koje su u poglavlju sa konvolucionim kodovima već opisane i ilustrovane primjerom. Najčešći oblik sistema za dekodiranje je Viterbijev algoritam, koji smo ranije upoznali, odnosno već smo se susreli s njegovom relativnom složenošću. Da bi situacija bila komplikovanija, ovdje moramo da kombinujemo, preko SISO procesora, rezultate dekodiranja dva konvoluciona koda, pa su stoga, da bi se izašlo u susret svim ovim izazovima, razvijene posebne varijante Viterbijevog algoritma za dekodiranje, kao što je npr. SOVA (engl. *Soft-Output-Viterbi-Algorthm*). U detalje ovog i većeg broja drugih algoritama koji su razvijeni za predmetnu namjenu nećemo ulaziti. Na Slici VII.15 prikazana je blok shema sistema za dekodiranje turbo-kodova.

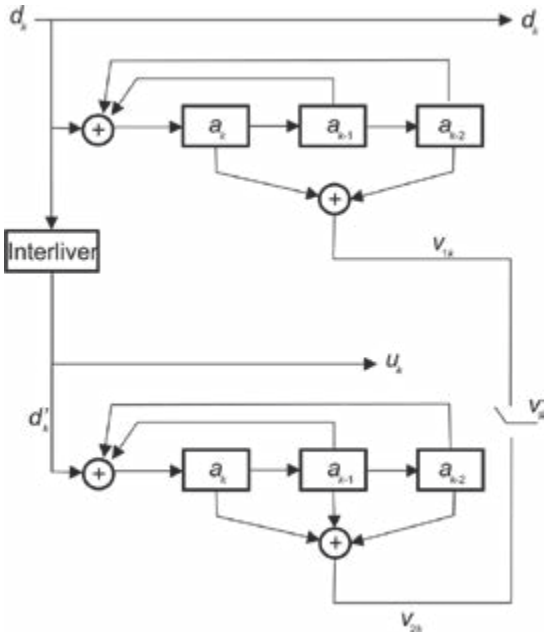
Ovim smo dali osnovne elemente funkcionisanja sistema za turbo-kodiranje i dekodiranje. Ovo je jedan od novijih postupaka u ovoj oblasti, koji prevazilazi blok i konvolucione kodove u smislu približavanja uslovima II Šenonove teoreme. Rad turbo-kodova, a posebno dekodiranje ovih kodova opisali smo na principskom nivou i dali jedan ilustrativni primjer za koji se nadamo da otkriva čitaocima kvalitete ovih kodova. Nadamo se da studenti koji pročitaju ovaj kratak materijal imaju dovoljno elemenata da nastave samostalno da izučavaju ovu materiju.

VII.4 Zadaci i softverska realizacija

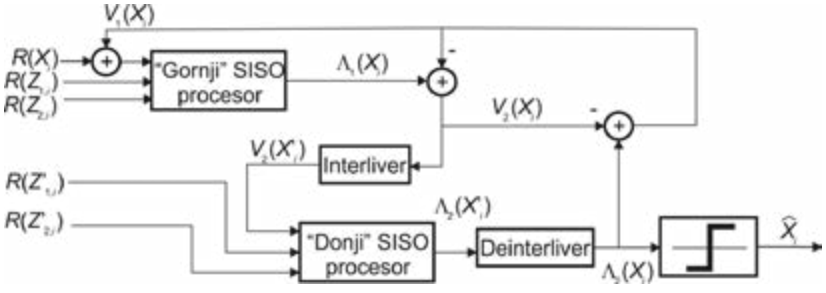
VII.4.1 Riješeni zadaci

7.1. Prikazati trelis dijagram za binarni Hemingov kod (7,4) sa generatorskim polinomom $\mathbf{p}(x)=x^3+x+1$ kojim se množi polinom koji predstavlja informacione bite.

Rješenje: Interesantan detalj je da se trelis šemom može prikazati kodiranje i dekodiranje i blok kodova. Predmetni kod ima tri stanja (Slika VII.16), tako da



Slika VII.14. Turbo-koder sa interliverom i dva rekurzivna sistematska konvoluciona kodera



Slika VII.15. Sistem za dekodiranje turbo-kodova zasnovan na SISO procesorima i SOVA algoritmu

su moguće vrijednosti, koje su upisane u ćelijama registra, 000, 001, 010, ..., 110, 111, odnosno ovo je 8 mogućih stanja. Na početku se polazi iz stanja 000. Iz ovog stanja može se preći u stanje 000 kada na ulazu dolazi bit 0 (isprekidana linija na Slici VII.17 označava prelaz indukovan nulom) sa izlazom 0 ili u stanje 100 (puna linija označava prelaz indukovan jedinicom).

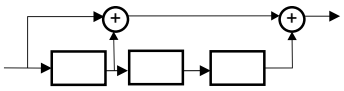
Prva tri koraka predstavljaju punjenje kodera informacionim bitima. Zbog toga trelis nije pun, odnosno ne postoje sva moguća stanja. U narednom koraku stiže četvrti informacioni bit i imamo puni trelis. Nakon četiri informaciona bita, koder se priprema za prijem narednog bloka, tako što prima tri nule, pa se trelis postepeno u tri koraka svodi na polazno stanje 000.

7.2. Izvršiti dekodiranje Hemingovog koda iz prethodnog primjera ako je poruka 1101101 primljena putem neke od tehnika koje su razvijene za konvolucione kodove.

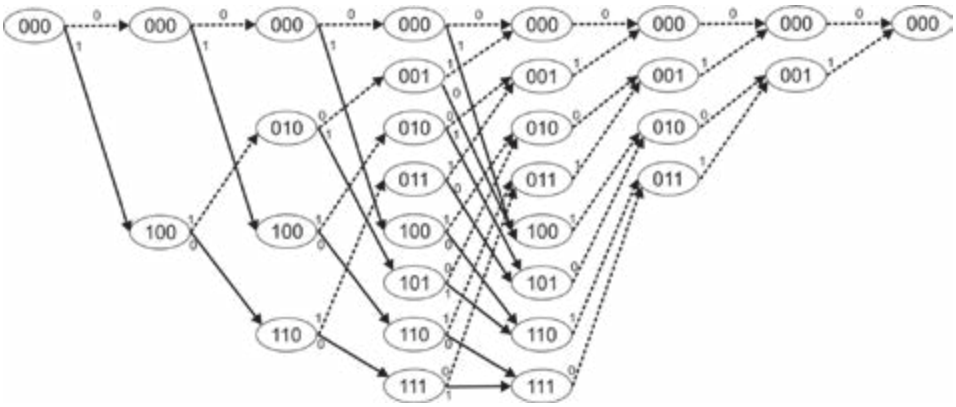
Rješenje: Lako je uočiti da predmetne putanje nema u trelisu, odnosno da u kodnoj riječi sigurno postoji pogreška. Ako pogledamo trelis sa Slike VII.17, ide se u donju granu, u prvom koraku, i prelazi u stanje 100, zatim se ide u stanje 010 (sa bitom 1 koji izlazi iz koda), pa u stanje 001, nakon toga u stanje 000, ali se iz tog stanja može preći samo u stanje 000, ali uz produkciju bita 0 na izlazu, a naša kodna riječ na petoj poziciji ima bit 1. Primijenimo sada Viterbijev algoritam (Slika VII.18). U uglastim zagradama upisane su Hemingove distance do pojedinih stanja. Zadržali smo najbolja parcijalna stanja (da smo imali s istim Hemingovim rastojanjem, zadržali bismo obje).

Na Slici VII.19 prikazana je putanja sa minimalnim Hemingovim rastojanjem. Lako je uočiti da se putanja razlikuje od primljene poruke 1101101 sa Hemingovim rastojanjem 1, a da se ta greška pojavila na četvrtom bitu, pa je ispravna kodna riječ 1100101. Vidimo da je ova sekvenca kodirana sa 1001 (pratimo prve četiri linije u trelisu pošto su naredne tri determinisane nulama i uočavamo da su prva i četvrta sa punom linijom – prouzrokovane jedinicom, dok su ostale dvije sa isprekidanom linijom, odnosno prouzrokovane nulom).

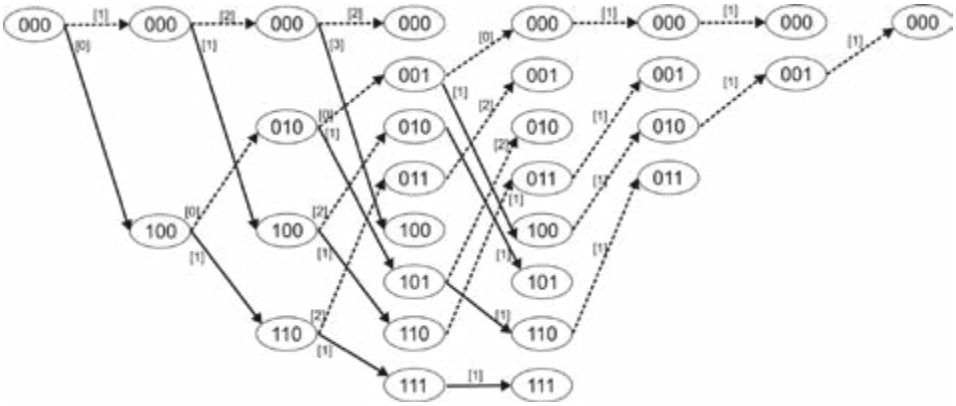
7.3. Za konvolucioni koder sa Slike VII.20: (a) Odrediti polinomijalni zapis funkcije ovog kodera, kao i matrični zapis preko matrice **G**. (b) Odrediti karak-



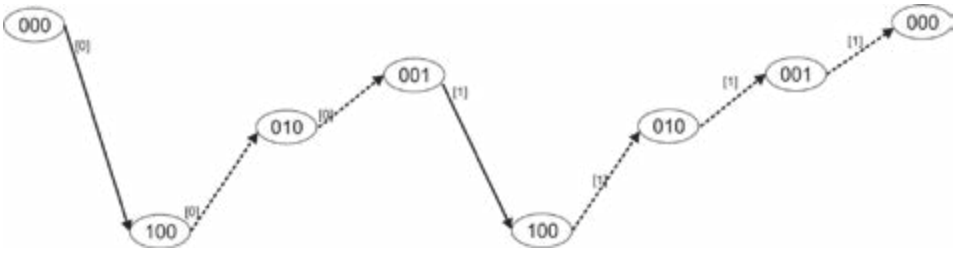
Slika VII.16. Koder predmetnog Hemingovog koda



Slika VII.17. Trelis dijagram za predmetni Hemingov kod



Slika VII.18. Trelis dijagram za primljenu kodnu riječ Hemingovog koda sa upisanim minimalnim Hemingovim rastojanjem za parcijalne najbolje putanje do pojedinih čvorova



Slika VII.19. Putanja sa minimalnim Hemingovim rastojanjem (dekodirana poruka)

teristike koda koji se formira zahvaljujući ovom sklopu (N, M, R, V itd.). (c) Formirati *look-up* tabelu sa 16 kolona u kojima su upisane vrijednosti ulaznog bita, početna stanja u ćelijama registra, na osnovu kojih treba sračunati naredna stanja u ćelijama registra, kao i izlazne bitove. (d) Nacrtati dijagram stanja ovog kodera. (e) Formirati trelis dijagram. (f) Kodirati poruku 1011 koristivši dijagram stanja, kao i pomoću trelisa (nemojte pretpostavljati da je trelis ograničen). (g) Pretpostaviti da je primljena poruka 01 11 01 11 0101 11. Ispraviti poruku (odrediti koliko je grešaka napravljeno i na kojim pozicijama). Dekodirati poslatu poruku. Prilikom kretanja kroz sekvencijalno stablo postaviti da se može ići u neku granu, dok je Hemingova distanca između primljene riječi manja od 3. Kad postane veća od 3, treba se vratiti na granu koja još nije provjerena, a koja ima Hemingovo rastojanje manje od 3. (h) Dekodiranje izvršiti Viterbijevim algoritmom. Uočite da ovaj kod ima 8 čvorova, odnosno 8 redova sa čvorovima u trelisu.

Rješenje: (a) U gornjoj grani se nalazi koder sa generatorskim polinomom $1+x+x^2+x^3$, dok je u donjoj grani koder sa generatorskim polinomom $1+x+x^3$.

Matrica **G** (uslovno je nazovimo generatorskom matricom) za predmetni koder ima oblik:

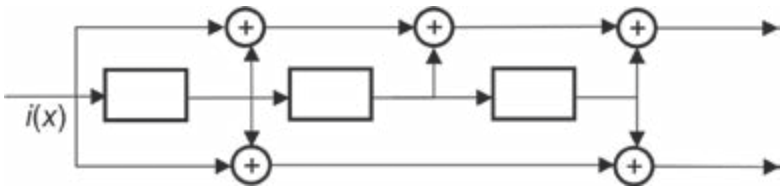
$$\mathbf{G} = [x^3 + x^2 + x + 1, x^3 + x + 1].$$

(b) Memorija ovog kodera je jednaka najvećem stepenu generatorskog polinoma, što je u ovom slučaju jednako $M = 3$. Ograničena dužina koda je $N = M + 1 = 4$. Kodni odnos ovog koda je $R = 1/2$ (broj vrsta kroz broj kolona generatorske matrice). Ukupna ograničena dužina koda je $V = 2 \times 4 = 8$ bita jer postoje dvije grane sa stepenima polinoma jednakim tri, odnosno sa 4 bita.

(c) *Look-up* tabela, koja prikazuje relaciju između stanja u registru i ulaznog bita sa izlazima iz kodera i novim stanjem, prikazana je u Tabeli VII.8.

(d) Dijagram stanja kodera prikazan je na Slici VII.21.

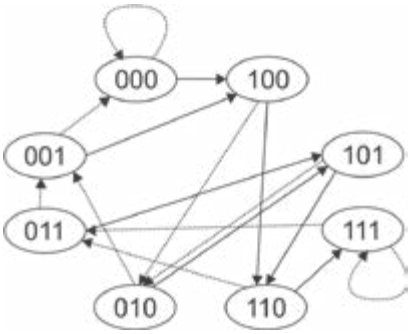
(e) Jedan korak u trelisu prikazan je na Slici VII.22. Svaka grana praćena je srednjim zagradama koje pokazuju izlaz iz kodera (respektivno izlazi iz gornjeg i donjeg kodera).



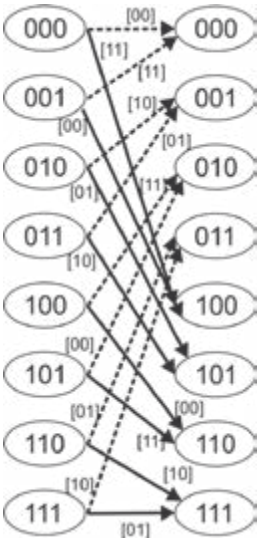
Slika VII.20. Konvolucioni koder iz zadatka 7.3

Ulazni bit	Stanje u ćelijama	Naredno stanje	Izlaz gore	Izlaz dolje
0	000	000	0	0
0	001	000	1	1
0	010	001	1	0
0	011	001	0	1
0	100	010	1	1
0	101	010	0	0
0	110	011	0	1
0	111	011	1	0
1	000	100	1	1
1	001	100	0	0
1	010	101	0	1
1	011	101	1	0
1	100	110	0	0
1	101	110	1	1
1	110	111	1	0
1	111	111	0	1

Tabela VII.8. Look-up tabela za konvolucioni kod iz zadatka 7.3



Slika VII.21. Dijagram stanja za koder iz zadatka 7.3, sa Slike VII.20



Slika VII.22. Jedan korak u trelisu za koder sa Slike VII.20.
(u srednjim zagradama prikazan je izlaz iz koda)

(f) Na Slici VII.23 prikazan je kompletan trelis za kod sa četiri bita i odgovarajućim ograničenjem. Sa ulaznom porukom 1011 dobija se putanja, odnosno kod označen podebljanom linijom 11 11 01 11 01 01 11.

(g) Za vježbu vam prepuštamo da putem sekvencijalnog dekodiranja izvršite dekodiranje ove sekvence.

(h) Mnogo atraktivniji postupak dekodiranja predstavlja Viterbijev postupak, pa smo stoga odlučili da samo njega prikažemo u okviru rješenja ovog zadatka. Na Slici VII.24 prikazan je postupak dekodiranja putem Viterbijevog algoritma. U prva tri koraka sačuvane su sve grane jer do svakog čvora stiže samo po jedna grana. Na par mjesta smo čuvali obje parcijalne najbolje putanje koje stižu do određenog

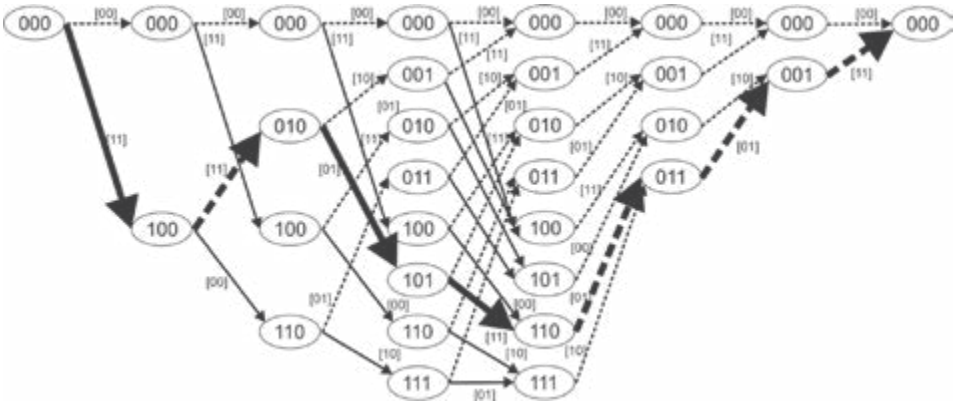
čvora pošto imaju istu težinu. Lako je uočiti da je poslata poruka 1011, te da se pogreška dogodila već na prvom bitu kodne riječi.

VII.4.2 Softverska realizacija

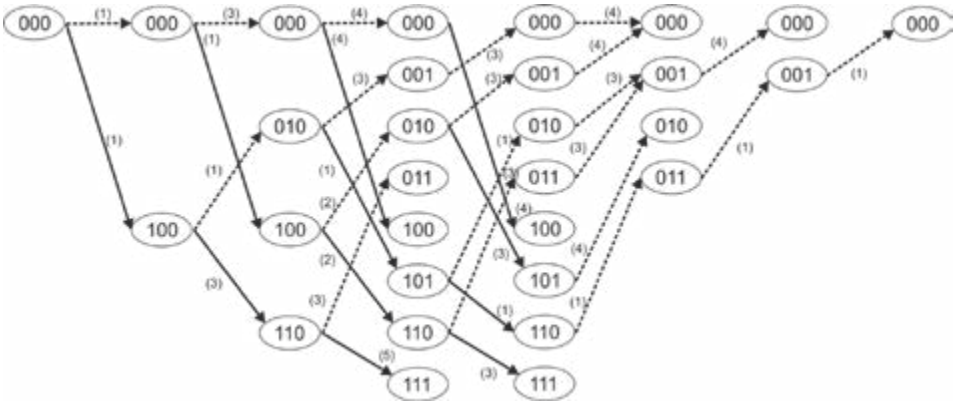
A. U novijim verzijama MATLAB-a postoji mogućnost rada sa konvolucionim kodovima. U srcu toga rada sa konvolucionim kodovima nalazi se funkcija poly2trellis. Funkcija ima na prvi pogled sasvim uobičajenu sintaksu:

```
T= poly2trellis(D,GEN);
```

Prvi argument, D, ove funkcije je ograničena dužina, dok je drugi, GEN, generatorski polinom zapisan u jednoj prilično neobičnoj formi. GEN ima onoliko članova koliko je grana sa izlazima iz kodera. Naime, kao što znamo, svaki generatorski polinom se može zapisati putem polinoma $\mathbf{g}(x) = g_0 + g_1 x + g_2 x^2 + ... + g_{n-1} x^{n-1}$, gdje



Slika VII.23. Trelis dijagram za koder iz zadatka 7.3 za ulaznu poruku 1011



Slika VII.24. Hemingove distance, najbolje parcijalne putanje i procjena dekodirane riječi

su kod binarnih kodova koeficijenti g_i biti 0 ili 1. Uočite da smo ređanje obavili od koeficijenta najmanjeg značaja. Na primjer, u kodu iz prethodnog zadatka generatorski polinom je prikazan na ovaj način:

$$\mathbf{G} = [1 + x + x^2 + x^3, 1 + x + x^3],$$

što se u binarnom zapisu može prikazati kao $\mathbf{G} = [1111, 1101]$. Međutim, kod funkcije poly2trellis zapisivanje se obavlja oktalno tako što sa kraja grupišemo po tri bita i prikazujemo ih putem oktalnog broja. Tako je GEN u ovom slučaju GEN=[17 15]. Kako je D=4 za naš primjer, odgovarajući trelis se (kao rezultat ove naredbe) dobija sa:

```
T=poly2trellis(4,[17 15])
T =
numInputSymbols: 2
numOutputSymbols: 4
numStates: 8
nextStates: [8x2 double]
outputs: [8x2 double]
```

Kao što vidimo, rezultat ove funkcije je struktura u kojoj su zapisani važni parametri našeg trelisa.

B. Funkcija za kodiranje konvolucionog koda je convenc. Ima samo dva parametra: prvi je niz bitova koji se kodiraju, a drugi je trelis. Formirajmo niz od 100 bitova, kodirajmo sekvencu, a zatim odredimo tri različita signala koji predstavljaju primljenu sekvencu sa 3% i 10% grešaka.

```
ix=randi([0 1],100,1);
cr=convenc(ix,T);
r1=xor(cr,rand(200,1)<0.03);
r2= xor(cr,rand(200,1)<0.1);
```

C. Dekodiranje se obavlja putem Viterbijevog dekodera, funkcijom vitdec. Ova funkcija ima pet argumenata: primljenu sekvencu, trelis, dubinu trelisa, operacioni mod dekodera i način donošenja odluke. Bez potrebe da dalje objašnjavamo, postavićemo da je operacioni mod dekodera 'trunc' (dekoder polazi od svih stanja sa nulama), te da se odluka donosi „tvrdim" odlučivanjem 'hard'. Dekodirajmo sve tri primljene poruke i uporedimo s informacionom sekvencom:

```
y1=vitdec(cr,T,34,'trunc','hard');
y2=vitdec(r1,T,34,'trunc','hard');
y3=vitdec(r2,T,34,'trunc','hard');
sum(xor(ix,y1))
    0
sum(xor(ix,y2))
```

```
    0
sum(xor(ix,y3))
    1
```

Vidimo da su u prva dva slučaja (signal bez pogreški i signal sa 3% pogreški) ispravljene sve greške, dok je u trećem slučaju, kada je primljena riječ zahvaćena sa 10% pogreški, „preživjela“ jedna greška.

D. Postoje takozvani katastrofalni konvolucioni kodovi. To su oni kod kojih, kada dekoder nije u stanju da ispravi pogrešku, dolazi do propagiranja pogreške i dekoder ne može više da je ispravi, odnosno poslije nekog vremena ne dolazi do stabilizacije rezultata i ispravnog dekodiranja. To možemo provjeriti sa:

```
iscatastrophic(T)
0
```

Dobili smo informaciju da je naš nekatastrofalan (što je dobro), dok bismo u suprotnom dobili rezultat 1. Ovim se ne zaokružuje programski sistem koji se u MATLAB-u može koristiti za rad sa konvolucionim kodovima, jer sve primijenjene, kao ni druge funkcije nisu objašnjene mnogo bogatije varijantama i mogu se primijeniti i u brojnim drugim situacijama. Suštinski, sistem ima mali broj funkcija, ali je veoma fleksibilan i pogodan za dalju dogradnju.

E. U trenutku pisanja ove knjige još uvijek ne postoji MATLAB-ov sistem za rad sa turbo-kodovima koji je približnog kvaliteta kao onaj za rad sa blok i konvolucionim kodovima. Preko sistema za razmjenu MATLAB fajlova (https://www.mathworks.com/matlabcentral/fileexchange/) dostupno je više alata, koje su razvili pojedinci i istraživačke grupe za turbo-kodiranje, pa su čitaoci slobodni da testiraju ove alate. Ipak, postoji sistem za rad sa specifičnim turbo-kodovima i to onima koji se koriste kod LTE (engl. *Long-Term Evolution*) telekomunikacionih sistema za mobilne komunikacije. Ovo su, suštinski, grupe kodova sa koderima prikazanim šematski na Slici VII.8. Funkcije za turbo-kodiranje i dekodiranje za ove kodove su lteTurboEncode i lteTurboDecode.

POGLAVLJE VIII

NAPREDNE TEME*

VIII. NAPREDNE TEME*

Udžbenik je, kao što smo već vidjeli, dominantno namijenjen onima koji se po prvi put sreću sa teorijom informacija i kodova, te onima koji imaju određena znanja pa im je potrebno njihovo produbljivanje i sistematizacija. Kao takav, na mnogim mjestima ne može da ide u dubinu za veliki broj postupaka i kodova koji se u praksi pojavljuju. Stoga je u ovom poglavlju rudimentarno obrađen određeni broj aktuelnih tema kako bi udžbenik dao uvid u razvoj ovih oblasti. Prvo ćemo obraditi granice koje se u procesu kodiranja mogu postići. Vidjeli smo da se u kanalima mogu desiti složeni obrasci grešaka, odnosno mogućnost pojavljivanja većeg broja uzastopnih grešaka. Problem detekcije ovakvih grešaka u praksi je (djelimično) riješen putem **CRC kodova** (kodovi sa cikličnom kontrolom redundancije, engl. *Cyclic Redundancy Check*). Zatim smo obradili **Rid−Mulerove kodove**. Da bi se mogli sagledati BCH kodovi i njihova generalizacija na nebinarne alfabete – **Rid−Solomonove kodove**, dali smo osnovne osobine Vandermondove matrice, koja se veoma često koristi kod praktičnih kodnih sistema. Naravno, nakon toga smo prikazali i same Rid−Solomonove kodove. Priča o savremenim kodovima ne bi bila kompletna kada se ne bismo osvrnuli na **LPDC kodove**, koji su se u praksi pojavili početkom ovog milenijuma i predstavljaju snažnu alternativu drugim kodnim postupcima, a prije svega turbo-kodovima. Ipak, teorijske osnove ovih kodova poznate su još iz 1960-ih. Konačno, nekoliko riječi ćemo kazati o **Golajevim kodovima**, koji su među prvim postupcima koji su primijenjeni za ispravljanje više pogreški. Poglavlje i samu knjigu zaključujemo pregledom pozicije i primjene učenih kodova u savremenim komunikacionim sistemima i kod nekih formata zapisa digitalnih podataka.

VIII.1 Granice kod kodova

Posmatrano s teorijske strane, granice za kodove predstavljaju bitne činjenice kod kodiranja kanala, do kojih se može doći bez razvoja partikularnih kodova. Ove granice pokazuju do čega se može doći kod kodova, a što je, zapravo, nemoguće. Ukazuju, na primjer, kolika je redundancija potrebna da bi se ostvarila određena mogućnost ispravljanja pogreški. Bilo bi zgodno da su granice što je moguće preciznije (uže), te da sugerišu na postupke konstrukcije kodova. Premda su glavni rezultati u ovoj oblasti poznati duži niz godina, ovo je i danas bogato polje istraživanja, sa novim rezultatima (koji neće biti domašeni u ovom kratkom pregledu) koji se pojavljuju relativno često. Sa prvom, **Hemingovom granicom** ili granicom **pakovanja sfera** već smo se upoznali, a ovdje je detaljnije elaboriramo. Premda Hemingova granica nije naročito precizna (kaže se − uska) popularna je jer se može lako shvatiti na osnovu jednostavne geometrijsko-kombinatorne analogije. Mi smo je i do sada koristili na primjeru binarnih kodova, a sada ćemo je generalizovati za slučaj opštih q-arnih kodova. Moguće je ukupno kreirati q^n riječi ovog kodnog alfabeta, dužine n. Hemingova težina $w_H(\mathbf{v})$ za opšti q-arni kod definiše se kao broj nenultih mjesta u kodnoj riječi: $\mathbf{v}=(v_1, ..., v_n)$, dok je Hemingova distanca $d_H(\mathbf{v},\mathbf{w})$ broj mjesta na kojima se kodne riječi $\mathbf{v}$ i $\mathbf{w}$ razlikuju. Veza Hemingove distance i težine je $w_H(\mathbf{v}-\mathbf{w})=d_H(\mathbf{v},\mathbf{w})$. Sfera radijusa r (mjereno Hemingovom distancom), u odnosu na posmatranu kodnu riječ $\mathbf{v}$, skup je svih vektora $\mathbf{w}$ za koje važi $d_H(\mathbf{v},\mathbf{w})\le r$. Zapreminom se naziva broj takvih vektora u sferi. Broj elemenata (zapremina sfere poluprečnika r) u posmatranom sistemu sa kodnim riječima dužine n je:

$$1+(q-1)\binom{n}{1}+(q-1)^2\binom{n}{2}+\cdots+(q-1)^r\binom{n}{r}.$$

Prvi član (broj 1) je sama kodna riječ $\mathbf{v}$, dok je sljedeći član broj kodnih riječi koje se razlikuju na jednoj poziciji od posmatrane kodne riječi. Broj pozicija je n, a za svaku poziciju postoji $(q-1)$ različitih vrijednosti u q-arnom alfabetu. Slično slijedi i za ostale članove u predmetnom redu.

Lema VIII.1. Neka su $\mathbf{x}$ i $\mathbf{y}$ dva vektora (kodne riječi) q-arnog alfabeta dužine n, za koje važi $d_H(\mathbf{x},\mathbf{y})>2e$, gdje je e cijeli broj. Ako za vektor $\mathbf{z}$ važi da je $d_H(\mathbf{x},\mathbf{z})\le e$, tada važi da je $d_H(\mathbf{y},\mathbf{z})>e$.

Dokaz. Dokaz je posljedica nejednakosti trougla: $d_H(\mathbf{x},\mathbf{y})\le d_H(\mathbf{x},\mathbf{z})+d_H(\mathbf{z},\mathbf{y})$ za bilo koju trojku vektora $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$. Ako je $d_H(\mathbf{x},\mathbf{y})>2e$ i $d_H(\mathbf{x},\mathbf{z})\le e$, tada slijedi da je: $2e<e+d_H(\mathbf{z},\mathbf{y})$, odnosno slijedi da je $e<d_H(\mathbf{z},\mathbf{y})$.

Posljedica VIII.1. Ako se vektori $\mathbf{x}$ i $\mathbf{y}$ nalaze na rastojanju koje je veće od $2e$, $d_H(\mathbf{x},\mathbf{y})>2e$, tada su sfere, koje su radijusa e, centrirane u $\mathbf{x}$ i $\mathbf{y}$ razdvojene.

Dokaz posljedice je prepušten čitaocima.

Teorema VIII.1. (Teorema o pakovanju sfera) Neka kod definisan nad q-arnim alfabetom, sa kodnim riječima dužine n ima minimalnu distancu $2e+1$ i l kodnih riječi. Tada važi:

$$l\left(1+(q-1)\binom{n}{1}+(q-1)^2\binom{n}{2}+\cdots+(q-e)^2\binom{n}{e}\right)\leq q^n.$$

Dokaz. Pošto je minimalna distanca $2e+1$, tada su, po nejednakosti trougla i na osnovu prethodne posljedice, bilo koje dvije sfere radijusa e, locirane u kodnim riječima, razdvojene (nemaju presjek). Dakle, suma zapremina l razdvojenih sfera biće manja ili jednaka ukupnoj zapremini čitavog skupa q^n.

Kod koji granicu pakovanja sfera dostiže sa jednakošću naziva se **perfektnim**. Postoji svega nekoliko perfektnih linearnih kodova: Hemingovi koji ispravljaju jednu grešku, dva Golajeva koda, kao i nekoliko nelinearnih kodova. Predmetna teorema stoga daje dosta široke granice, ali je fizikalnost njenog tumačenja takva da se ona obično jedino i uči na mnogim osnovnim kursevima teorije informacija i kodova.

Druga granica koju ćemo ovdje uvesti je **Gilbert−Varšamova** (engl. *Edgar Gilbert*, rus. Ром Рубенович Варшамов), koja se izvodi u suprotnom pravcu u odnosu na Hemingovu. Ona pokušava da dokaže postojanje linearnog koda sa datim parametrima (uočite da je ograničena na linearne kodove). Ova teorema takođe ne daje veoma usku granicu za formiranje kodova. Posmatramo sada linearni blok kod na alfabetu sa q-elemenata, dužine n, dimenzije (broja informacionih simbola) k i sa minimalnom distancom d. Generatorska matrica ovog koda dimenzija je $k\times n$. Za potrebe predmetne analize označimo ovaj kod kao $(n,k)_d$, gdje smo minimalnu distancu koda stavili u indeks uobičajenog označavanja kod blok kodova.

Teorema VIII.2. Linearni (blok) kod $(n,k)_d$, definisan preko alfabeta sa q simbola, postoji ako je zadovoljeno:

$$q^{n-k}-1>(q-1)\binom{n-1}{1}+\cdots+(q-1)^{d-3}\binom{n-1}{d-3}+(q-1)^{d-2}\binom{n-1}{d-2}.$$

Radi jednostavnosti, teoremu dokazujemo za binarni slučaj koji se svodi na:

$$2^{n-k}-1>\binom{n-1}{1}+\binom{n-1}{2}+\cdots+\binom{n-1}{d-3}+\binom{n-1}{d-2},$$

dok čitaocima prepuštamo generalizaciju dokaza za opšti q-arni alfabet. Prije samog dokaza napomenimo da ova procedura ukazuje samo na očekivane performanse

koda. Međutim, u nekim slučajevima moguć je dizajn boljih kodova, a sa druge strane granica ne potvrđuje da kodovi sa navedenim performansama postoje.

Dokaz. Pretpostavimo da smo u konstrukciji kontrolne matrice koda s uspjehom izabrali l kolona, tako da je $l \geq d-2$ i da su $l-1$ linerno zavisne. Sada želimo da izaberemo $(l+1)$ kolonu. Izbor se mora napraviti između vektor kolona dužine $n-k$. Takvih postoji 2^{n-k}. Isključujemo kolone sa svim nulama, sve kolone koje su se prethodno pojavljivale, sumu svih kombinacija prethodne dvije kolone i tako sve do sume bilo koje $d-2$ prethodne kolone. U najgorem slučaju, svi elementi koji se „izbacuju" moraju biti različiti. Stoga, dostupnih vektora može da bude samo:

$$2^{n-k} - \left(1 + \binom{l}{1} + \binom{l}{2} + \cdots + \binom{l}{d-2}\right).$$

Da bi predmetna selekcija bila dostupna ovaj broj mora biti pozitivan. Uzmimo granični slučaj $l = d-2$. Moralo bi biti zadovoljeno:

$$2^{n-k} \geq 1 + \binom{d-2}{1} + \binom{d-2}{2} + \cdots + \binom{d-2}{d-2}.$$

Desnu stranu možemo da zapišemo korišćenjem binomnog obrasca, kao:

$$2^{d-2} = (1+1)^{d-2} = \sum_{i=0}^{d-2} \binom{d-2}{i} 1^i 1^{d-2-i} = \sum_{i=0}^{d-2} \binom{d-2}{i},$$

pa slijedi:

$$2^{n-k} \geq 2^{d-2}$$
$$n-k \geq d-2,$$

a ovo je zasigurno zadovoljeno pošto kolone kontrole matrice moraju da imaju dužinu od najmanje $d-1$. Kako raste l, binomni koeficijenti $\binom{l}{i}$ su rastući, pa to znači da ako prethodna nejednakost važi za granični slučaj, važi i za sve prethodne (sa manjim l), pa konstatujemo da smo postigli i dokazali uslove teoreme.

Gilbert−Varšamova nejednakost ukazuje da možemo kreirati kod sa parametrima $(n, k)_d$ ako je odgovarajuća nejednakost zadovoljena. Nažalost, do sada nije nađena nijedna procedura, osim direktnog pretraživanja, koja bi konstruisala predmetni kod. Suprotan stav od Gilbert−Varšamove teoreme nije tačan. Naime, možda postoji linearni kod koji prevazilazi limite ove teoreme. Do danas je nađeno više takvih kodova, kao što su, na primjer, (geometrijski) Gopa (rus. Вале́рий Дени́сович Го́ппа) kodovi. Trebalo je da prođe oko 30 godina od uvođenja Gilbert−Varšamove granice (1952) do pronalaska prvih kodova koji su u stanju da je prevaziđu.

Singletonova granica (engl. *Richard Collom Singleton*) se može primijeniti na bilo koji kod. Primjenjuje se da bi pokazala nepostojanje određenog koda, a ne da bi pokazala postojanje koda sa datim parametrima.

Teorema VIII.3. Dat je kod sa kodnom riječju dužine n, definisan nad q-arnim alfabetom, s minimalnom distancom d i l kodnih riječi: Tada je:

$$q^{n-(d-1)} \geq l.$$

Napomena. Ako je kod linearan, na primjer (n,k) kod, tada je broj kodnih riječi $l = q^k$ i, uzimajući logaritam osnovne q, Singletonova granica postaje:

$$n-(d-1) \geq k,$$

što se alternativno može zapisati kao:

$$n+1 \geq k+d.$$

Dakle, ako je $k+d \leq n$ tada ne postoji odgovarajući kod.

Dokaz. Pošto se svaki par kodnih riječi razlikuje na najmanje d pozicija, čak i ako ignorišemo posljednjih $d-1$ pozicija, nema dvije kodne riječi takve da će biti iste u prvih $n-(d-1)$ kodnih bita. Ako otkinemo posljednjih $d-1$ pozicija, svakih l kodnih riječi su i dalje različite. Dakle, dobićemo kod sa l kodnih riječi i dužinom bloka $n-(d-1)$. Pošto postoji $q^{n-(d-1)}$ q-arnih riječi dužine $n-(d-1)$, slijedi da je $l \leq q^{n-(d-1)}$.

Kodovi koji postižu predmetnu granicu sa jednakošću nazivaju se kodovi sa maksimalnom distancom razdvajanja (engl. *Maximum Distance Separating*) ili kraće MDS kodovima.

U praksi su definisane brojne druge granice za kodove, kao što su, na primjer: Plotkinova, Džonsonova (engl. *Johnson*), Grizmerova (engl. *Griesmer*) itd.

VIII.2 CRC Kodovi

CRC (engl. *Cyclic Redundancy Check*) kodovi su klasa kodova koji su prevashodno namijenjeni za detekciju većeg broja grešaka. CRC kodovi se kreiraju na osnovu generatorskog polinoma čiji koeficijenti uzimaju vrijednosti iz nekog konačnog polja. Najčešće je to polje binarno. Posebno je značajna sposobnost CRC kodova da detektuju pojavu više grešaka u relativno dugačkim kodnim riječima. Pretpostavimo da je generatorski polinom ovog koda $\mathbf{g}(x)$. Neka je $\mathbf{i}(x)$ polinom koji predstavlja informaciju, dok je $\mathbf{c}(x)$ polinom koji predstavlja kodnu riječ. Tokom prenosa u kanalu, došlo je do distorzije (promjene) kodne riječi, te je primljena poruka $\mathbf{r}(x)$. Greška u prenosu se može izraziti kao:

$$\mathbf{e}(x)=\mathbf{c}(x)\text{-}\mathbf{r}(x),$$

gdje upotrijebljeni operator ima odgovarajući smisao u odnosu na upotrijebljeni alfabet (npr. kod binarnog alfabeta riječ je o operaciji ekskluzivno ili). Broj nenultih koeficijenata (Hemingova težina) $\mathbf{e}(x)$ je broj pogreški. Definišimo $\mathbf{o}(x)$ ostatak pri dijeljenu $\mathbf{c}(x)$ sa generatorskim polinomom koda $\mathbf{g}(x)$. Ovaj ostatak se često naziva CRC-om od kodnog polinoma $\mathbf{c}(x)$. Sada možemo pisati:

$$\mathbf{c}(x)=\mathbf{q}(x)\mathbf{g}(x)+\mathbf{o}(x).$$

Kada u kodu postoje greške, umjesto količnika $\mathbf{q}(x)$ i ostatka $\mathbf{o}(x)$, dobijamo drugačije vrijednosti, koje možemo označiti kao $\tilde{\mathbf{q}}(x)$ i $\tilde{\mathbf{o}}(x)$. Stoga možemo zapisati:

$$\begin{aligned}\mathbf{e}(x)&=\mathbf{c}(x)-\mathbf{r}(x)=(\mathbf{q}(x)\mathbf{g}(x)+\mathbf{o}(x))-(\tilde{\mathbf{q}}(x)\mathbf{g}(x)+\tilde{\mathbf{o}}(x))=\\&=(\mathbf{q}(x)-\tilde{\mathbf{q}}(x))\mathbf{g}(x)+\mathbf{o}(x)-\tilde{\mathbf{o}}(x).\end{aligned}$$

Ostatak pri dijeljenju $\mathbf{e}(x)$ sa $\mathbf{g}(x)$ jednak je $\mathbf{o}(x)-\tilde{\mathbf{o}}(x)$. CRC ne radi ispravno kada je ova razlika jednaka 0, što je ekvivalentno da $\mathbf{e}(x)$ dijeli $\mathbf{g}(x)$ bez ostatka.

Ako se greška dogodi samo na jednom bitu, tada se polinom greške može zapisati kao $\mathbf{e}(x)=x^i$. Greška se ne može detektovati samo u slučaju kada $\mathbf{g}(x)$ dijeli $\mathbf{e}(x)=x^i$ bez ostatka. Pošto je x^i proizvod i kopija od x, $\mathbf{g}(x)$ može da dijeli x^i samo ako je $\mathbf{g}(x)$ jednako x^j za neko $j\le i$. Dakle, čak i najjednostavniji polinom koji je različit od stepena x, kao što je $\mathbf{g}(x)=x+1$, sposoban je za detekciju jedne greške. U pitanju je polinom za provjeru parnosti koji se koristi kod ASCII koda. Stoga se može zaključiti da je detekcija jedne pogreške jednostavna. Ako postoje dvije pogreške na m-toj i na n-toj poziciji (sa $m<n$), tada je polinom greške:

$$\mathbf{e}(x)=x^m+x^n.$$

Do pogreške u detekciji sa CRC kodom dolazi samo ako generatorski polinom $\mathbf{g}(x)$ dijeli $\mathbf{e}(x)=x^m+x^n$. Polinom $\mathbf{e}(x)$ možemo faktorisati kao:

$$\mathbf{e}(x)=x^m+x^n=x^m(1+x^{n-m}).$$

Relativno je lako obezbijediti da $\mathbf{g}(x)$ ne dijeli x^m, ali nije baš jednostavno analizirati da dizajn $\mathbf{g}(x)$ ne dijeli bilo koju varijantu $1+x^{n-m}$. Uočimo da važi:

$$x^N-1=(x-1)(x^{N-1}+x^{N-2}+\ldots+x+1)$$

za bilo koje N. Slično važi ako zamijenimo, recimo, x sa x^8, pa se može zapisati:

$$x^{16}-1=(x^8-1)(x^8+1)$$

$$x^{24}-1=(x^8-1)(x^{16}+x^8+1)$$

$$x^{32}-1=(x^8-1)(x^{24}+x^{16}+x^8+1)$$

$$x^{40}-1=(x^8-1)(x^{32}+x^{24}+x^{16}+x^8+1)$$
$$\ldots$$
$$x^{8N}-1=(x^8-1)(x^{8(N-1)}+\ldots+x^8+1)\ .$$

Dakle, uočavamo da predmetni kod, zasnovan na generatorskom polinomu x^8-1 (odnosno x^8+1 zbog osobine operacije ex-ili), neće detektovati dvije greške ako se one nalaze na rastojanju koje je umnožak od 8. Predmetna osobina može biti postignuta i sa polinomima manjih dimenzija. Npr. CRC sa generatorskim polinomom x^3+x+1, iako je samo trećeg stepena, a ne osmog, daje veoma sličnu osobinu, odnosno ne detektuje dvije pogreške samo ako su ove pogreške razmaknute za broj cifara koji je umnožak broja 7. Dakle, x^3+x+1 dijeli x^N-1 sa ostatkom 0 samo ako je N množilac broja 7. Ova osobina je u direktnoj vezi s upotrebom ovog prostog polinoma kod kreiranja Hemingovog koda (7,4). Slično važi da x^4+x+1 prosti polinom ne može da detektuje dvije greške samo kada je rastojanje između pozicija na kojima su se pogreške dogodile jednak umnošku broja 15. Koristivši iskustva iz Hemingovih kodova, prosti polinom x^5+x^2+1 ne može da detektuje dvije greške samo kada su na rastojanju koje je umnožak broja 31. Treba znati da ne može svaki polinom petog stepena da ponovi ovako kvalitetan rezultat. Tako polinom x^5+x+1 može da otkrije greške koje nisu razmaknute za broj cifara koji je umnožak 21, dok polinom x^5+x^4+x+1 ne može da detektuje pogreške koje su razmaknute za umnožak od 8 bitova. Jedan izuzetno moćan CRC kod može se dobiti generatorskim polinomom:

$$\mathbf{g}(x)=x^{16}+x^{15}+x^2+1=(x+1)(x^{15}+x+1).$$

Ovaj kod je u stanju da detektuje dvije pogreške osim ako se nalaze na rastojanju koje je umnožak broja $2^{15}-1$ (32767). Dakle, dobri generatorski polinomi za CRC kodove moraju biti prosti (nesvodivi) polinomi ili njihovi proizvodi. Druga osobina je da dati polinom mora biti djelilac x^N-1, gdje je $N=2^d-1$, a d je stepen predmetnog polinoma. Napomenimo dalje da je $\mathbf{g}(x)$ takav generatorski polinom da je u stanju da detektuje bilo koju kombinaciju od 3 greške u setu podataka koji je kraći od $2^{15}-1$ (32767). Razlog je u činjenici da se kao faktor kod ovog polinoma pojavljuje polinom $x+1$ koji je u stanju da uvijek detektuje neparan broj grešaka (prva osobina vezana za dvije greške jedino je vezana za prosti polinom $x^{15}+x+1$, a nema veze sa $x+1$).

Pored mogućnosti detekcije nekoliko pogreški, kod CRC kodova važna je i osobina da se pomoću njih može detektovati i nekoliko uzastopnih pogreški (engl. *burst errors*). CRC kod reda n uvijek je u stanju da detektuje uzastopne greške koje imaju dužinu koja je kraća od n. Da bismo ovo dokazali, zapisaćemo polinom pogreške kao:

$$\mathbf{e}(x)=x^n\mathbf{p}(x),$$

gdje je $\mathbf{p}(x)$ polinom reda koji je manji od n. CRC ne bi bio u stanju da detektuje ovakav tip pogreške u slučaju da $\mathbf{g}(x)$ dijeli $\mathbf{e}(x)$. Jasno je da $\mathbf{g}(x)$ nije djelilac od x^n, a ujedno, pošto je $\mathbf{p}(\boldsymbol{x})$ manjeg reda od n, $\mathbf{g}(x)$ ne može podijeliti ni $\mathbf{p}(x)$.

Na ovaj način smo u kratkim crtama opisali osnovne osobine CRC kodova.

VIII.3 Rid–Mulerovi kodovi

Rid–Mulerovi (engl. *Reed–Muller*) kodovi predstavljaju proširenje standardnih Hemingovih kodova. Razvijeni su 1954. godine, nezavisno od dvojice pomenutih naučnika. Danas imaju ograničen značaj u teoriji kodova, ali se intenzivno koriste u teoriji složenosti i u nekim drugim disciplinama. Za nas su ovi kodovi interesantan primjer procesa proširivanja kodova, odnosno konstrukcije jedne klase kodova na osnovu druge klase koja ima dobro utemeljenje. Biće opisani samo binarni Rid–Mulerovi kodovi prvog reda. Dužina kodne riječi kod ovih kodova je dijadična, odnosno jednaka stepenu dvojke $n = 2^p$, gdje je p cijeli broj. Generatorska matrica dijadičnog Rid–Mulerovog koda opisuje se rekurzivno:

$$\mathrm{RM}_p = \left[\begin{array}{c|c} 00\ldots0 & 11\ldots1 \\ \hline \mathrm{RM}_{p-1} & \mathrm{RM}_{p-1} \end{array}\right],$$

gdje je inicijalna generatorska matrica formirana kao:

$$\mathrm{RM}_1 = \left[\begin{array}{c|c} 0 & 1 \\ \hline 1 & 1 \end{array}\right].$$

Svaka naredna se formira tako što se doda jedna vrsta sa 2^p nula i 2^p jedinica nakon dvije generatorske matrice manjih dimenzija:

$$\mathrm{RM}_2 = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right] \qquad \mathrm{RM}_3 = \left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$\mathrm{RM}_4 = \left[\begin{array}{cccccccc|cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right].$$

Matrica RM_3 ima jednu vrstu sa svim jedinicama. Matrični minor koji čine prve tri vrste i kolone od druge nadalje sadrži sve različite nenulte trobitne riječi. Zaklju-

čujemo da je ovo kontrolna matrica proširenog Hemingovog (8,4) koda. Iz generatorske matrice ovog koda lako se zaključuje da je dužina kodne riječi 2^p (koliko ima vrsta u riječi), dok je broj informacionih simbola $p+1$ (koliko ima vrsta ove matrice). Dakle, riječ je o kodovima s ekstremno velikom dužinom kodne riječi, ali, s druge strane, Rid−Mulerovi kodovi imaju veoma veliko minimalno Hemingovo rastojanje. Naime, minimalno Hemingovo rastojanje kod ovog koda je 2^{p-1}. Ovo se može dokazati na više načina. Npr. prvi kod RM_1 ima dva informaciona bita i dva bita u kodnoj riječi i vrši samo permutaciju informacionih bita na određeni način. Minimalno Hemingovo rastojanje takvog jednoznačno dekodabilnog koda, koji ne može da obezbijedi nikakvu provjeru, jednako je jedan. Minimalno Hemingovo rastojanje za kod sa više bitova jednako je sumi minimalnog Hemingovog rastojanja na pojedinačnim bitima. Kod Rid−Mulerovih kodova uvijek imamo dvije grupe bitova. Na primjer, za RM_2 imamo kod (4,3), tako da kod prva dva bita informacione riječi dobijamo RM_1 kod od drugog i trećeg bita informacione riječi. Dakle, prva dva bita u kodnoj riječi imaju minimalno Hemingovo rastojanje 1. Na isti način se dobija da druga dva bita imaju minimalno rastojanje 1 jer su jednaki prvom informacionom bitu i RM_1 kodu računatom za preostala dva informaciona bita. Dakle, RM_2 ima minimalno Hemingovo rastojanje $1+1=2$. Za svaki naredni Rid−Mulerov kod minimalno Hemingovo rastojanje jednako je dvostrukom minimalnom Hemingovom rastojanju prethodnog koda, tako da dokazujemo da je minimalno Hemingovo rastojanje Rid−Mulerovog koda 2^{p-1}. Veliko minimalno Hemingovo rastojanje za veliko p ukazuje na dobru sposobnost ispravljanja grešaka.

Rid−Mulerovi kodovi se koriste i u bipolarnoj formi. Kod bipolarne forme svaka nula u kodnoj riječi mijenja se sa – 1. Skalarni proizvod dvije ispravne kodne riječi ovog koda tada zadovoljava sljedeću osobinu: skalarni proizvod je suma proizvoda simbola na odgovarajućim pozicijama. Skalarni proizvod dvije ovakve kodne riječi u tom slučaju ima vrijednost koja je jednaka broju pozicija na kojima se pojavljuju isti predznaci i umanjena za broj pozicija sa različitim predznacima. Ovaj produkt se može zapisati i kao $n-2d_H(\mathbf{x},\mathbf{y})$, gdje je n dužina koda, a $d_H(\mathbf{x},\mathbf{y})$ Hemingovo rastojanje dvije kodne riječi (broj pozicija na kojima se kodne riječi razlikuju). Skalarni proizvod za ispravne kodne riječi RM koda može se sračunati kao:

$$\mathbf{c}_i \cdot \mathbf{c}_j = \begin{cases} \pm 2^p & \mathbf{c}_i = \pm \mathbf{c}_j \\ 0 & \mathbf{c}_i \neq \pm \mathbf{c}_j . \end{cases}$$

Dakle, dobijamo izuzetno velike vrijednosti (posebno za veće matrice) ako vršimo dekodiranje odgovarajuće kodne riječi (ili njoj inverzne), dok u ostalim slučajevima za sve ostale kodne riječi dobijamo vrijednost koja je jednaka nuli. U slučaju da postoje greške u kodu, umjesto $\pm 2^p$ i 0, kao rezultat ovog postupka dobićemo

različite vrijednosti. Međutim, pravilnim postavljanjem praga (engl. *threshold*) detekcije, posebno za veće p, možemo da izvršimo ispravno dekodiranje.

Još interesantnija posljedica ovog postupka je njegova prilagođenost na meko odlučivanje. Naime, predmetni postupak je veoma pogodan da umjesto sa bitovima (ili njihovom bipolarnom varijantom) radimo sa stvarno primljenim mekim vrijednostima (npr. ako vrijednosti 1 odgovara napon od 5V, mjereni napon, neka je to 3.7V, poredimo s nekim pragom, npr. 2.5V, i odlučujemo da smo primili jedan ili nula). Međutim, veći broj savremenih kodova i dekodirajućih algoritama (što smo vidjeli kod turbo-kodova) radi sa mekim (decimalnim) vrijednostima u procesu dekodiranja, pošto je očigledno vrijednost 5.5V sigurnija jedinica nego je to vrijednost 3.7V.

Dekodiranje se u ovom domenu može obaviti na osnovu Adamardove (fr. *Hadamard*) transformacije. Adamardova transformacija se definiše rekurzivno (na sličan način kao matrica Rid−Mulerovog koda):

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \mathbf{H}_4 = \begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_2 \\ \mathbf{H}_2 & -\mathbf{H}_2 \end{bmatrix} \qquad \mathbf{H}_{2^n} = \begin{bmatrix} \mathbf{H}_{2^{n-1}} & \mathbf{H}_{2^{n-1}} \\ \mathbf{H}_{2^{n-1}} & -\mathbf{H}_{2^{n-1}} \end{bmatrix}.$$

Za Adamardovu matricu važi:

$$\mathbf{H}_m \mathbf{H}_m^T = m\mathbf{I}_m,$$

gdje je $\mathbf{I}_m$ kvadratna matrica dimenzija $m \times m$. Pretpostavimo da je primljena sekvenca $\mathbf{r} = [1, 1, -1, 1, -1, -1, 1, 1]$. Za dekodiranje ove poruke nam je potrebna Adamardova matrica dimenzija 8×8:

$$\mathbf{H}_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

Dekodiranje se obavlja množenjem transformacione matrice sa primljenom porukom:

$$\mathbf{H}_8 \mathbf{r}^T = \begin{bmatrix} 2 & -2 & 2 & -2 & 2 & -2 & 6 & 2 \end{bmatrix}^T.$$

Kako je najveća apsolutna vrijednost dobijena na poziciji 7, možemo zaključiti da ispravna kodna riječ odgovara sedmoj koloni (ili vrsti, jer je Adamardova matrica simetrična):

$$\mathbf{c}_7^8 = [1 \quad 1 \quad -1 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1].$$

U slučaju da je najveća apsolutna vrijednost u Adamardovoj transformaciji primljenog vektora negativna, neophodno je primljenu kodnu riječ zamijeniti njenom negacijom (promijeniti predznak svim elementima). Pretpostavimo da smo primili poruku u kojoj su elementi, zbog fizičkih djelovanja u kanalu, izmijenjeni. Neka je ta poruka:

$$\mathbf{r} = \begin{bmatrix} -0.7 & 1 & 0 & -0.8 & -0.9 & 1 & 0.9 & -1 \end{bmatrix}.$$

Dekodiranje izvršimo na osnovu predmetne „meke poruke" koja nije poređenjem sa pragom pretvorena u bite (u našem slučaju bipolarne ±1). Adamardova transformacija je u ovom slučaju:

$$\mathbf{H}_8\mathbf{r}^T = \begin{bmatrix} -0.5 & -0.9 & 1.3 & -6.3 & -0.5 & -0.9 & 0.9 & 1.3 \end{bmatrix}^T.$$

Najveću apsolutnu vrijednost sada ima četvrti član niza, ali kako je negativan, zaključujemo da negacija četvrte kolone (vrste) Adamardove matrice predstavlja odgovarajuću kodnu riječ:

$$-\mathbf{c}_4^* = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

Zbog primjene Adamardove transformacione matrice kod se često i naziva Adamardovim. Najčešće se primjenjuje na riječi dijadične dužine (stepena dvojke), ali može i na neke druge. Predmetno dekodiranje za koje se koriste nebinarni, odnosno decimalni podaci naziva se mekim i slično je viđenom kod turbo-kodova. Dobra strana primjene Adamardove matrice u dekodiranju je izuzetna efikasnost množenja kontrolne (Adamardove) matrice sa dobijenim vektorom. Naime, u opštem slučaju, množenje matrice dimenzija $n \times n$, čiji su elementi ±1 sa vektorom dužine n, zahtijeva $n \times (n-1)$ sabiranja (zanemarimo za trenutak množenje sa 1 i promjenu znaka). Međutim, postoji mogućnost da se postigne značajna ušteda u broju potrebnih operacija. Naime, operaciju množenja vektora $\mathbf{r}$, dužine n, sa Adamardovom matricom možemo zapisati preko dva vektora $\mathbf{r}_1$ i $\mathbf{r}_2$, od kojih vektor $\mathbf{r}_1$ predstavlja prvih $n/2$ elemenata vektora $\mathbf{r}$, dok je $\mathbf{r}_2$ vektor preostalih $n/2$ odbiraka vektora $\mathbf{r}$:

$$\mathbf{H}_n\mathbf{r}^T = \begin{bmatrix} \mathbf{H}_{n/2} & \mathbf{H}_{n/2} \\ \mathbf{H}_{n/2} & -\mathbf{H}_{n/2} \end{bmatrix} \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{n/2}\mathbf{r}_1^T + \mathbf{H}_{n/2}\mathbf{r}_2^T \\ \mathbf{H}_{n/2}\mathbf{r}_1^T - \mathbf{H}_{n/2}\mathbf{r}_2^T \end{bmatrix}.$$

Izrazi $\mathbf{H}_{n/2}\mathbf{r}_1^T$ i $\mathbf{H}_{n/2}\mathbf{r}_2^T$ predstavljaju Adamardove transformacije odgovarajućih vektora koji imaju dužinu $n/2$. Za proračun svake od ovih Adamardovih transformacija bilo je potrebno po $(n/2)(n/2-1)$ sabiranja, tako da je za proračun Adamardove transformacije na ovaj način ukupno potrebno:

$$n(n/2-1)+n \text{ sabiranja.}$$

Ovo je, zapravo, 2 puta koliko je potrebno za Adamardovu transformaciju po $n/2$ članova plus n dodatnih sabiranja za svaki elemenat. Dakle, ako bismo ovo primijenili na Adamardovu transformaciju vektora dužine $n=8$, bilo bi nam potrebno 32 sabiranja, dok je standardnim algoritmom potrebno 56 sabiranja. Ako nastavimo sa daljim dekomponovanjem, za Adamardovu matricu dimenzija $n/2$ potrebno je $n^2/8$ operacija, dok je za Adamardovu transformaciju dimenzija $n/4$ potrebno $n^2/32$ operacije. Pretpostavimo da je za n elemenata vektora potrebno $f(n)$ operacija, a za $n/2$ potrebno $f(n/2)$ operacija. Veza između broja operacija za n elemenata i $n/2$ elemenata može se opisati kao:

$$f(n)=2f(n/2)+n.$$

Funkcija koja zadovoljava ovu relaciju je $f(n)=n\log_2 n$. Dakle, za $n=8$ ako je Adamardova dekompozicija vršena „do kraja“, odnosno do matrica dimenzija 2×2, dobija se da je potrebno 24 operacije, što je značajna ušteda. Npr. za $n=1024$ operacije, umjesto preko milion operacija, potrebno je samo nešto više od 10 hiljada.

VIII.4 Vandermondova matrica

U teoriji kodova često se koristi Vandermondova matrica. Ova matrica je dobro poznata konstrukcija iz linearne algebre, ali ćemo je ovdje kratko ponovo razmotriti pošto se u teoriji kodova definiše preko elemenata iz konačnih polja. Vrijedi reći da se Vandermondova matrica koristi u teoriji i praksi kod razvoja BCH, Rid−Solomonovih i srodnih kodova.

Poznato je da linearni kod može da ispravi e pogreški ako je $2e+1$ kolona njegove kontrolne matrice linearno nezavisno. Još u linearnoj algebri smo učili da su l vektora dužine l linearno nezavisni ako korespondirajuća matrica ima determinantu različitu od nule. Nažalost, računanje determinante nije jednostavno, pa nam treba neki trik koji bi nam ukazivao da je neka klasa matrica nesingularna (sa determinantom različitom od nule). Ovdje ćemo predstaviti dva tipa Vandermondovih matrica koje zadovoljavaju ove osobine pod veoma lakoprovjerljivim uslovima. Posmatrajmo n vektora, dužine n, koji čine matricu:

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & \cdots & v_{1n} \\ v_{21} & v_{22} & v_{23} & v_{24} & \cdots & v_{2n} \\ v_{31} & v_{32} & v_{33} & v_{34} & \cdots & v_{3n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & v_{n3} & v_{n4} & \cdots & v_{nn} \end{bmatrix}.$$

Jedan od oblika Vandermondove matrice za koji se pouzdano zna da produkuje nenultu determinantu je Vandermondova matrica koja je nesingularna kada je svako $x_i \neq x_j$ za $i \neq j$:

$$\mathbf{VM} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & x_4 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & x_4^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}.$$

Determinanta ove matrice je takođe poznata i ima oblik:

$$\det(\mathbf{VM}) = (-1)^{n(n-1)/2} \prod_{i<j} (x_i - x_j).$$

Na osnovu elementarne matematike znamo da je prozvod nenultih elemenata različit od nule, ali ovo ne mora da važi za konačna polja, već je osobina koju zadovoljavaju integralni domeni, kao što je, recimo, skup cijelih brojeva.

Posmatrajmo polje od k konačnih elemenata koji će biti stepeni od α: 1, α, α^2, α^3, ..., α^l. Onda važi:

$$\det \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & (\alpha^3)^2 & \cdots & (\alpha^{n-1})^2 \\ 1 & \alpha^3 & (\alpha^2)^3 & (\alpha^3)^3 & \cdots & (\alpha^{n-1})^3 \\ 1 & \alpha^4 & (\alpha^2)^4 & (\alpha^3)^4 & \cdots & (\alpha^{n-1})^4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & (\alpha^2)^{n-1} & (\alpha^3)^{n-1} & \cdots & (\alpha^{n-1})^{n-1} \end{bmatrix} \neq 0.$$

Nenulti elementi se mogu preurediti na bilo koji pogodan poredak α^{l_1}, α^{l_2}, α^{l_3}, ..., α^{l_n}, pa i tada važi:

$$\det\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha^{l_1} & \alpha^{l_2} & \alpha^{l_3} & \cdots & \alpha^{l_{n-1}} \\ 1 & (\alpha^{l_1})^2 & (\alpha^{l_2})^2 & (\alpha^{l_3})^2 & \cdots & (\alpha^{l_{n-1}})^2 \\ 1 & (\alpha^{l_1})^3 & (\alpha^{l_2})^3 & (\alpha^{l_3})^3 & \cdots & (\alpha^{l_{n-1}})^3 \\ 1 & (\alpha^{l_1})^4 & (\alpha^{l_2})^4 & (\alpha^{l_3})^4 & \cdots & (\alpha^{l_{n-1}})^4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (\alpha^{l_1})^{n-1} & (\alpha^{l_2})^{n-1} & (\alpha^{l_3})^{n-1} & \cdots & (\alpha^{l_{n-1}})^{n-1} \end{bmatrix} \neq 0.$$

Ako se redovi ili vrste nesingularne matrice pomnože sa nenultim vrijednostima, matrica ostaje nesingularna. Ova osobina se koristi da bi se odredila druga forma Vandermondove matrice, koja se dobija tako što se svaka vrsta matrice pomnoži redom sa x_1, x_2, ..., x_n:

$$\mathbf{M} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & \cdots & x_n^2 \\ x_1^3 & x_2^3 & x_3^3 & x_4^3 & \cdots & x_n^3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & x_3^n & x_4^n & \cdots & x_n^n \end{bmatrix}.$$

Već smo se uvjerili da za svaki blok kod možemo koristiti kontrolnu matricu i to s elementima preko proizvoljnog polja $\mathbf{F}_q$. Svaki ciklični blok kod dužine n može se prikazati preko generatorskog polinoma $\mathbf{g}(x)$, koji je djelilac $x^n - 1$. Kolekcija svih kodnih riječi specificiranih sa $\mathbf{g}(x)$ je kolekcija svih polinoma stepena $< n$, koji su polinomi množioci $\mathbf{g}(x)$. Kod (n,k) ima osobinu $k = n - \deg(\mathbf{g})$ i $\mathbf{g}(x) = c_0 + c_1 x + \ldots + c_s x^s$. Generatorska matrica koda može se zapisati kao:

$$\mathbf{G} = \begin{bmatrix} c_0 & c_1 & \cdots & c_s & 0 & 0 & \cdots & 0 \\ 0 & c_0 & \cdots & c_{s-1} & c_s & 0 & \cdots & 0 \\ 0 & 0 & \cdots & c_{s-2} & c_{s-1} & c_s & \cdots & 0 \end{bmatrix}.$$

Neka je:

$$\mathbf{h}(x) = \frac{x^n - 1}{\mathbf{g}(x)} = b_0 + b_1 x + \ldots + b_t x^t \quad (t + s = n).$$

Jedan tip kontrolne matrice može se zapisati kao:

$$\mathbf{H} = \begin{bmatrix} b_t & b_{t-1} & b_{t-2} & b_{t-3} & \cdots & b_1 & b_0 & 0 & \cdots & 0 \\ 0 & b_t & b_{t-1} & b_{t-2} & \cdots & b_2 & b_1 & b_0 & \cdots & 0 \\ 0 & 0 & b_t & b_{t-1} & \cdots & b_3 & b_2 & b_1 & \cdots & 0 \\ & & \cdots & & & & & & & \end{bmatrix}.$$

Sada se mogu kreirati različiti oblici kontrolnih matrica koje bolje ilustruju sposobnosti kodova za ispravljanje grešaka preko linearne nezavisnosti kolona.

VIII.5 Rid–Solomonovi kodovi

Veoma popularna klasa kodova su Rid−Solomonovi (RS) kodovi (engl. *Reed−Solomon codes*). U osnovi definisanja Rid−Solomonovih kodova nalazi se Vandermondova matrica. Rid−Solomonovi kodovi se definišu preko polja sa prostim brojem elemenata. Pretpostavimo da imamo q simbola, gdje je q prost broj $\{0, 1, \ldots, q-1\}$. Neka je a generatorski element polja čiji stepeni generišu sve različite nenulte elemente polja: $\{1, a, a^2, \ldots, a^{q-3}, a^{q-2}\}$, $a^l \neq 1$ za $l < q-1$ i $a^{q-1} = 1$, $a^m \neq a^r$ za $m \neq r$, $m, r < q-1$. Može se pokazati da važi:

$$x^{q-1} - 1 = (x-a)(x-a^2)\cdots(x-a^{q-3})(x-a^{q-2}).$$

Formirajmo polinom reda $n-k$ (k je broj informacionih bita, dok je n dužina kodne riječi):

$$\mathbf{g}(x) = (x-a)(x-a^2)\cdots(x-a^{n-k-1})(x-a^{n-k}).$$

Kod koji je specificiran sa ovim generatorskim polinomom je zasigurno cikličan jer $\mathbf{g}(x)$ dijeli $x^{q-1} - 1$. Blok kod (n,k), definisan preko polja dimenzija q, naziva se Rid−Solomonov. Minimalna distanca ovog koda je definisana kao $d = n - k + 1$. Ponekad se ova minimalna distanca (ili dizajnirana distanca) kod ovih kodova označava kao t. Razlog za ovakvu minimalnu distancu potiče iz činjenice što se svake dvije ispravne kodne riječi moraju razlikovati na makar jednoj poziciji. Kada se takvi polinomi množe generatorskim polinomom, onda se rezultujući polinomi moraju razlikovati na makar $n-k+1$ poziciji. Ako je $d = t = 2e+1$, Rid−Solomonov kod je u stanju da ispravi e pogreški.

Posmatrajmo primjer polinoma koji je konstruisan nad poljem sa sedam elemenata, $q = 7$, i generatorskim polinom četvrtog reda sa nulom, $a = 4$:

$$\begin{aligned}\mathbf{g}(x) &= (x-3)(x-3^2)(x-3^3)(x-3^4) = \\ &= (x-3)(x-2)(x-6)(x-4) = x^4 + 6x^3 + 3x^2 + 2x + 4.\end{aligned}$$

Predmetni kod će imati dimenziju (6,2) jer je $n-k=4$ (što je očigledno iz reda generatorskog polinoma), pa, uzimajući da je $k=2$, dobijamo da je $n=6$. Distanca predmetnog koda je $d=5$, pa može da ispravi dvije pogreške. Generatorska matrica je dimenzija $k \times n = 2 \times 6$ i formira se koeficijentima generatorskog polinoma (počev od onoga uz nulti stepen) i popunjavanjem preostalih mjesta nulama:

$$\mathbf{G} = \begin{bmatrix} 4 & 2 & 3 & 6 & 4 & 0 \\ 0 & 4 & 2 & 3 & 6 & 4 \end{bmatrix}.$$

Kontrolna matrica se sljedstveno može dobiti tako što se odredi polinom $\mathbf{h}(x)$ koji kod cikličnih kodova zadovoljava:

$$\mathbf{h}(x)=\frac{x^6-1}{\mathbf{g}(x)}=(x-1)(x-5)=x^2+x+5.$$

Dimenzije kontrolne matrice su $(n-k)\times n=4\times 6$, koja se dobija na sličan način, ali sada uz reverzan redosljed koeficijenata:

$$\mathbf{H}=\begin{bmatrix}1&1&5&0&0&0\\0&1&1&5&0&0\\0&0&1&1&5&0\\0&0&0&1&1&5\end{bmatrix}.$$

Predmetni kod je dužine 6, može da ispravi dvije pogreške i ima kodni odnos $R=2/6=1/3$.

Primjer VIII.1. Kreirati Rid−Solomonov kod koji može da koriguje tri pogreške.

Rješenje: Minimalna distanca koda mora biti 7, a kako $d=t\le q-1$, možemo usvojiti $q=11$ kao najmanji prosti broj koji ovo zadovoljava. Generator polja $\{1, 2, ..., 11\}$ može biti $a=2$, pa se generatorski polinom $t-1=$ šestog reda može zapisati kao:

$$\begin{aligned}\mathbf{g}(x)&=(x-2)(x-2^2)(x-2^3)(x-2^4)(x-2^5)(x-2^6)=\\&=x^6+9x^5+9x^4+7x^3+x^2+4x+8.\end{aligned}$$

Jasno je da polinom $\mathbf{h}(x)$ sada ima nule koje su preostale nule polinoma $x^{10}-1$:

$$\mathbf{h}(x)=(x-2^7)(x-2^8)(x-2^9)(x-1)=x^4+5x^3+9x^2+2x+5.$$

Naravno, važi da je:

$$\mathbf{g}(x)\mathbf{h}(x)=x^{10}-1.$$

Generatorska matrica ovog koda ima dimenziju 4×10 (može se formirati na osnovu generatorskog polinoma):

$$\mathbf{G}=\begin{bmatrix}8&4&1&7&9&9&1&0&0&0\\0&8&4&1&7&9&9&1&0&0\\0&0&8&4&1&7&9&9&1&0\\0&0&0&8&4&1&7&9&9&1\end{bmatrix}.$$

Kontrolna matrica koda ima dimenzije 6×10 i može se formirati na osnovu polinoma $\mathbf{h}(x)$ (ponovo obratite pažnju da se **G** formira na osnovu $\mathbf{g}(x)$, sa koeficijentima u rastućem redosljedu po pripadajućim stepenima, dok je **H** formirana na osnovu $\mathbf{h}(x)$, sa koeficijentima u opadajućem redosljedu po pripadajućim stepenima):

$$\mathbf{H} = \begin{bmatrix} 1 & 5 & 9 & 2 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 5 & 9 & 2 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 5 & 9 & 2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 5 & 9 & 2 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 5 & 9 & 2 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 5 & 9 & 2 & 5 \end{bmatrix}.$$

Može se pokazati da je svaki minor koji se sastoji od 6 kolona matrice **H** međusobno nezavisan (rang matrice je 6), pa predmetni (10,4) kod može korigovati 3 pogreške (čemu smo se nadali kada smo usvajali dizajniranu distancu koda). Kodni odnos predmetnog koda sa alfabetom definisanom na skupu od $q = 11$ simbola je $R = 4/10$. □

Kodni odnos kod Rid−Solomonovih kodova može se izraziti na različite načine:

$$R = \frac{n - \deg \mathbf{g}}{n} = \frac{q-d}{q-1} = 1 - \frac{d-1}{q-1} = 1 - \frac{2w}{q-1},$$

gdje je w broj grešaka koje kod može da ispravi. Jasno je da se mora uspostaviti odgovarajući kompromis između kodnog odnosa i sposobnosti koda da ispravlja greške.

Rid−Solomonovi kodovi se mogu dekodirati putem Euklidskog algoritma. Vrijedi istaći da postoji niz drugih postupaka za predmetnu svrhu, koji su često popularniji, mada manje jednostavni za objasniti, iz teorijskih, računskih, hardverskih i softverskih razloga. Primjena Euklidskog dekodirajućeg algoritma na dekodiranje Rid−Solomonovih kodova biće prikazana kroz primjer.

Primjer VIII.2. Dat je kod sa generatorskim polinomom $\mathbf{g}(x) = x^4 + 6x^3 + 3x^2 + x^2 + 4$, definisanim preko polja sa $q = 7$ simbola. Primljena poruka je $\mathbf{r}(x) = 4x^5 + 5x^4 + 5x^3 + 3x + 1$. Ispraviti primljenu riječ i izvršiti dekodiranje poruke.

Rješenje: Očigledno se radi o kodu dužine 6, gdje je broj informacionih simbola 2, koji je u stanju da ispravi dvije pogreške. Već smo vidjeli u prvom poglavlju da $a = 3$ može da posluži kao generator predmetnog polja. Koeficijente sindromskog polinoma određujemo kao:

$$S_i = \mathbf{r}(a^i), \text{ za } i \in [1,4].$$

Odgovarajuće vrijednosti su:

$$S_1 = \mathbf{r}(a^1) = 3 \qquad S_2 = \mathbf{r}(a^2) = 3$$
$$S_3 = \mathbf{r}(a^3) = 1 \qquad S_4 = \mathbf{r}(a^4) = 4.$$

Dakle, sindromski polinom ima oblik:

$$\mathbf{S}(x) = 4x^3 + x^2 + 3x + 3.$$

Tabela VIII.1 prikazuje korake u Euklidskom algoritmu provedenom nad $x^{2w} = x^4$ i sindromskim polinomom. U ovom slučaju nam je, pored polinoma lokatora greške $\mathbf{v}(x) = 3x^2 + 2x + 6$ iz kolone **V**, bitan i polinom vrednovanja pogreške iz kolone **U**, $\mathbf{u}(x) = 2x + 3$. Nule polinoma lokatora greške su:

$$x_1 = 5 = 3^5 \qquad x_2 = 6 = 3^3.$$

Dakle, zaključujemo da su se pogreške dogodile na trećoj i petoj poziciji u kodnoj riječi, čitano od početka, na isti način kao što smo imali kod BCH kodova. Ostaje pitanje određivanja težina pogreške. Postoji manje ili više složenih postupaka da se odredi težina greške. Ovdje ćemo upotrijebiti Fornijev postupak (engl. *Forney* po *George David Forney jr.*, koji je dao brojne doprinose u komunikacijama i teoriji informacija). Prvi korak u ovom postupku je da se polinom lokator pogreške preskalira dijeljenjem koeficijentom uz član najnižeg reda u polinomu (ovdje je to 6):

$$\mathbf{l}(x) = \mathbf{v}(x)/6 = 4x^2 + 5x + 1.$$

Zatim odredimo izvod ovog polinoma (računanje se obavlja na način koji je za određivanje izvoda uobičajen, imajući na umu polje u kome se izračunavanje vrši):

$$\mathbf{l}'(x) = 8x + 5 = x + 5.$$

Pomnožimo polinom lokator greške $\mathbf{l}(x)$ sa sindromskim polinomom:

$$\mathbf{l}(x)\mathbf{S}(x) = 2x^5 + 3x^4 + 4x + 3.$$

Odredi se ostatak pri dijeljenju ovog polinoma sa x^4 (polinoma koji je korišćen prilikom provođenja Euklidskog algoritma), odnosno poništimo sve članove polinoma stepena koji je veći ili jednak 4, čime dobijamo:

K	Q	R	U	V
-1	...	x^4	1	0
0	...	$4x^3 + x^2 + 3x + 3$	0	1
1	$2x + 3$	$5x^2 + 6x + 5$	1	$5x + 4$
2	$5x + 4$	$3x + 4$	$2x + 3$	$3x^2 + 2x + 6$

Tabela VIII.1. Euklidski algoritam za dekodiranje Rid-Solomonovog koda iz Primjera VIII.2

$$\Omega(x) = 4x + 3.$$

Težine pogrešaka se dobijaju kao:

$$\mathbf{e}(x_1) = -\Omega(x_1)/\mathbf{l}'(x_1) = -\Omega(5)/\mathbf{l}'(5) = -23/10 = -2/3 = 5/3 = 4$$

jer je $4 \times 3 = 5$. Na isti način se odredi težina pogreške na drugoj lokaciji:

$$\mathbf{e}(x_2) = -\Omega(x_2)/\mathbf{l}'(x_2) = -\Omega(6)/\mathbf{l}'(6) = -27/11 = -6/4 = 1/4 = 2,$$

što znači da je težina pogreške na drugoj lokaciji 2. Oduzimanjem odgovarajućih težina na korespondentnim mjestima, dobijamo kodni polinom (dodajemo nulti član uz x^2 radi jasnoće):

$$\mathbf{c}(x) = \mathbf{r}(x) - \mathbf{e}(x) = 4x^5 + 5x^4 + (5-2)x^3 + 0x^2 + (3-4)x + 1 = 4x^5 + 5x^4 + 3x^3 + 0x^2 + \\ + (-1)x + 1 =$$

$$4x^5 + 5x^4 + 3x^3 + 0x^2 + 6x + 1.$$

Dekodirajmo poslatu poruku dijeljenjem sa generatorskim polinomom:

$$\mathbf{i}(x) = \mathbf{c}(x)/\mathbf{g}(x) = (4x^5 + 5x^4 + 3x^3 + 0x^2 + 6x + 1)/(x^4 + 6x^3 + 3x^2 + x^2 + 4) = 4x + 2.$$

Dakle, simboli koji su poslati u kanal bili su 4 i 2.

VIII.6 LDPC kodovi

Nije rijedak slučaj da se pojedine ideje koje budu u nekom vremenu odbačene u nauci poslije mnogo godina pojave kao novo, spasonosno rješenje, sposobno da prevaziđe određene probleme koje do tada nijedan drugi postupak nije bio u stanju.

Često su ograničenja koja su bila uspostavljena u ovom ili onom razdoblju protokom vremena prevaziđena i rješenja i postupci koji u to doba nisu mogli praktično da se implementiraju ponovo se pojavljuju na sceni. Tri su česta razloga za ovakva dešavanja: razvoj računarske industrije, prije svega, hardvera, koji nam omogućava da efikasno realizujemo određene postupke koje prije neku dekadu nismo bili u stanju realizovati; novi teorijski progres; drugačiji kontekst u koji se neki od postupaka stavlja.

Ova sudbina je zadesila i kodove sa provjerama parnosti niske gustine (LDPC kodovi – engl. *low-density parity-check codes*). Njihov nastanak se može pratiti još od 1960-ih godina, kada ih je uveo Galager (engl. *Robert G. Gallager*), radeći na doktorskoj disertaciji na MIT-u. Ponekad se, u njegovu slavu, zovu i Galagerovi kodovi. Iz samog imena kodova slijedi da oni nisu ništa drugo do kodovi sa provjerama parnosti, kao i svi kodovi koje smo do sada uvodili! Ne postoji prepreka da, na primjer, Hemingove kodove posmatramo kao specijalni slučaj LDPC kodova.

Tri uslova koja treba da zadovolji kod da bi se smatrao **regularnim LDPC kodom** (pored onih koje smo do sada uveli) su:

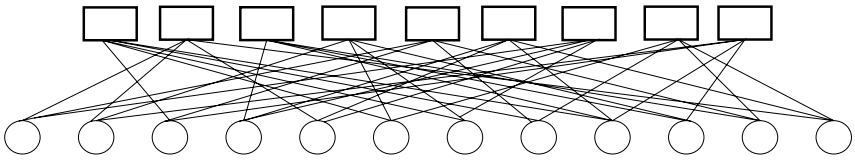
- broj jedinica u kontrolnoj matrici mora biti mali;
- svaka kolona kontrolne matrice ima isti broj jedinica (svaka cifra se onda pojavljuje u istom broju provjera parnosti);
- svaka vrsta kontrolne matrice ima isti broj jedinica.

Postoje i iregularni LDPC kodovi, kod kojih su prethodni drugi i/ili treći uslov donekle relaksirani. Prvi uslov je donekle neodređen, odnosno postavlja se pitanje šta to znači mali broj jedinica u kontrolnoj matrici. Posmatrajmo sljedeću kontrolnu matricu koda (12,3) (sa 9 provjera parnosti):

$$\mathbf{H}=\begin{bmatrix}0&0&1&0&0&1&1&1&0&0&0&0\\1&1&0&0&1&0&0&0&0&0&0&1\\0&0&0&1&0&0&0&0&1&1&1&0\\0&1&0&0&0&1&1&0&0&1&0&0\\1&0&1&0&0&0&0&1&\mathbf{0}&0&1&0\\0&0&0&1&1&0&0&0&1&0&0&1\\1&0&0&1&1&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&1&0&0&1&1\\0&1&1&0&0&0&0&0&1&1&0&0\end{bmatrix}$$

Kao što vidimo, u svakom redu kontrolne matrice postoje po četiri bita koja se provjeravaju, a u svakoj koloni postoje tri bita parnosti. To znači da se svaki bit u kodnoj riječi provjerava tri puta. LDPC kodovi se često prikazuju putem Tenerovog grafa (dobio ime po Majklu Teneru, engl. *Michael Tanner*). U Tenerovom grafu pojavljuju se dva skupa čvorova. Ovi čvorovi se obično vizuelizuju na različite načine.

Prvi tip čvorova predstavlja broj provjera parnosti (na Slici VIII.1 ovi čvorovi su prikazani putem pravougaonika), dok drugi tip čvorova (prikazan putem kružnica) predstavlja bite u kodnoj riječi. Stoga se predmetni graf sastoji od $(n-k)+n$ čvorova. Veze u grafu predstavljaju jedinice u grafu, odnosno bite koji učestvuju u pojedinim provjerama parnosti. Na Slici VIII.1 prikazan je graf koji predstavlja



Slika VIII.1. Tenerov graf za LDPC kod dat kontrolnom matricom iz primjera

prethodno uvedenu kontrolnu matricu. Kao i kod svih složenih kodnih postupaka, problem LDPC kodova nije u procesu kodiranja, već u procesu dekodiranja. Postupak dekodiranja LDPC kodova je zasnovan na razmjeni (prosljeđivanju) poruka među čvorovima i granama grafa.

Poruke koje se razmjenjuju u procesu dekodiranja predstavljaju vjerovatnoće (ili neku drugu ekvivalentnu mjeru) da je ispravan bit u kodnoj riječi jednak jedinici. Na čvorovima se ove informacije kombinuju na neki odgovarajući način. Za svaki bit postupak započinje s određenom inicijalnom vjerovatnoćom, dobijenom na osnovu mekog ulaza u koder, da je taj bit jednak jedinici. Zatim, ovaj čvor prenosi tu informaciju linijama do pozicija gdje se provjeravaju parnosti (čvorovi označeni pravougaonicima na Slici VIII.1). Dalje, čvorovi prave nove estimacije za bitove i vraćaju ih nazad ka bitovima (čvorovima prikazanim kružićima). Svaki od čvorova parnosti zna da treba da ima paran broj jedinica, ali umjesto toga on je snabdjeven vjerovatnoćama da su pojedini bitovi jedinice i na osnovu dobijenih informacija pravi se nova estimacija pojedinih bitova. Posmatrajmo sada prvu provjeru parnosti, u kojoj učestvuju biti c_3, c_6, c_7 i c_8. Pretpostavimo da su procjene vjerovatnoća da su ova četiri bita jednaka 1 respektivno date kao p_3, p_6, p_7 i p_8. Ažurirana vjerovatnoća p_3 se može sračunati kao:

$$p_3' = p_6(1-p_7)(1-p_8) + p_7(1-p_6)(1-p_8) + p_8(1-p_6)(1-p_7) + p_6 p_7 p_8.$$

Pojasnimo ovaj izraz. Naime bit c_3 biće jednak 1 ako su $\{c_6 = 1,\ c_7 = 0,\ c_8 = 0\}$, $\{c_6 = 0, c_7 = 1, c_8 = 0\}$, $\{c_6 = 0, c_7 = 0, c_8 = 1\}$ ili $\{c_6 = 1, c_7 = 1, c_8 = 1\}$, što prethodni izraz jasno pokazuje. Na sličan način se mogu ažurirati i ostale tri vjerovatnoće. Međutim, svaki čvor sa bitom (kružić) sada prima više različitih informacija o vjerovatnoći da je jednak 1, na osnovu provjera parnosti sa kojima je povezan. On ne šalje samo jednu vrijednost dalje, već različite procjene za svaku provjeru parnosti. Ova nova estimacija koja se šalje prema čvorovima dobijena je na osnovu svih drugih estimacija. Dakle, u određivanju nove estimacije koja se šalje prema čvoru parnosti ignoriše se estimacija koju je taj čvor parnosti poslao prema tom čvoru bita.

Po pravilu, ova nova estimacija koja se šalje prema čvoru parnosti jednaka je normalizovanom proizvodu drugih estimacija. Način normalizacije će biti objašnjen kasnije. Napomenimo da se obično ne šalje vjerovatnoća simbola, već logaritamski odnos vjerovatnoća da je neki bit jednak jedinici, odnosno nuli:

$$\log[p_i / (1-p_i)].$$

Estimacija kanala se uvijek koristi u svim procjenama koje se prosljeđuju ka čvorovima parnosti. Pretpostavimo da imamo čvor koji je uključen u tri provjere parnosti. Neka je procjena vjerovatnoće dobijena iz kanala p_K, dok neka su p^j,

$j = 1, 2, 3$ procjene dobijene za dati čvor na osnovu tri čvora parnosti u kojima on učestvuje. Nove estimacije koje se šalju prema čvorovima parnosti su:

$$p'^{j} = kp_K \prod_{\substack{i=1 \\ i \neq j}}^{3} p^{j}, \quad j = 1,2,3,$$

gdje je k parametar koji će biti objašnjen kasnije. Proces se dalje ponavlja. Vjerovatnoće se prenose ka čvorovima bitova, a čvorovi bitova dalje prenose vjerovatnoće ka bitovima parnosti.

U posljednjem koraku dobija se konačna procjena za svaki čvor, računanjem normalizovanog proizvoda svih njegovih estimacija. Konačna estimacija dobijena je za svaki čvor računanjem normalizovanih proizvoda svojih estimacija kao (za slučaj čvora koji učestvuje u tri provjere parnosti):

$$p'^{j} = kp_K \prod_{i=1}^{3} p^{j}, \quad j = 1,2,3.$$

Primjer VIII.3. Pretpostavimo sljedeći primjer, vezan za razmatrani kod sa Slike VIII.1. Pretpostavimo da smo iz kanala dobili sljedeće vjerovatnoće da su posmatrani bitovi jednaki 1:

$$0.9\ 0.5\ 0.4\ 0.3\ 0.9\ 0.9\ 0.9\ 0.9\ 0.9\ 0.9\ 0.9\ 0.9,$$

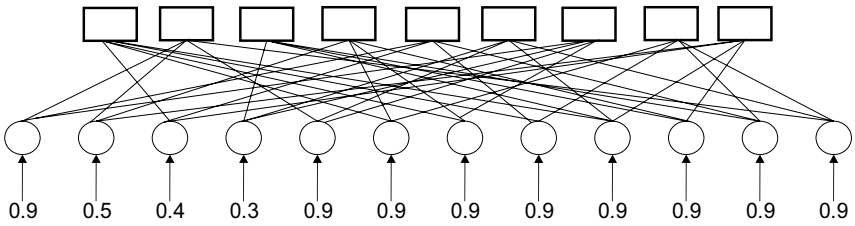
što je demonstrirano na Slici VIII.2. Prvi čvor parnosti provjerava treći, šesti, sedmi i osmi bit, čime se dobijaju ažurirane vjerovatnoće:

$$p_3^{(1)} = p_6(1-p_7)(1-p_8) + p_7(1-p_6)(1-p_8) + p_8(1-p_6)(1-p_7) + p_6p_7p_8 =$$
$$= 0.9 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.9 \cdot 0.9 = 0.756$$

$$p_6^{(1)} = p_3(1-p_7)(1-p_8) + p_7(1-p_3)(1-p_8) + p_8(1-p_3)(1-p_7) + p_3p_7p_8 =$$
$$= 0.4 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.6 \cdot 0.1 + 0.9 \cdot 0.6 \cdot 0.1 + 0.4 \cdot 0.9 \cdot 0.9 = 0.432$$

$$p_7^{(1)} = 0.432 \qquad p_8^{(1)} = 0.432.$$

Eksponentom smo indicirali da se radi o prvom čvoru parnosti.



Slika VIII.2. Tenerov graf sa ulaznim vjerovatnoćama u sistem

U drugom čvoru parnosti (drugi red matrice **H**) učestvuju biti sa pozicija 1, 2, 5 i 12, tako da su korespondentne vjerovatnoće:

$$p_1^{(2)} = p_2(1-p_5)(1-p_{12}) + p_5(1-p_2)(1-p_{12}) + p_{12}(1-p_2)(1-p_5) + p_2p_5p_{12} =$$
$$= 0.5 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.5 \cdot 0.1 + 0.9 \cdot 0.5 \cdot 0.1 + 0.9 \cdot 0.5 \cdot 0.9 = 0.5$$

$$p_2^{(2)} = p_1(1-p_5)(1-p_{12}) + p_5(1-p_1)(1-p_{12}) + p_{12}(1-p_1)(1-p_5) + p_1p_5p_{12} =$$
$$= 0.9 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.5 \cdot 0.9 = 0.432$$

$$p_5^{(2)} = p_1(1-p_2)(1-p_{12}) + p_2(1-p_1)(1-p_{12}) + p_{12}(1-p_1)(1-p_2) + p_1p_2p_{12} =$$
$$= 0.9 \cdot 0.5 \cdot 0.1 + 0.5 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.1 \cdot 0.5 + 0.9 \cdot 0.5 \cdot 0.9 = 0.5$$

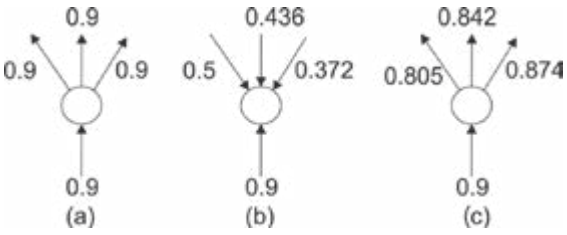
$$p_5^{(2)} = p_1(1-p_2)(1-p_{12}) + p_2(1-p_1)(1-p_{12}) + p_{12}(1-p_1)(1-p_2) + p_1p_2p_{12} =$$
$$= 0.9 \cdot 0.5 \cdot 0.1 + 0.5 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.1 \cdot 0.5 + 0.9 \cdot 0.5 \cdot 0.9 = 0.5$$

$$p_{12}^{(2)} = p_1(1-p_2)(1-p_5) + p_2(1-p_1)(1-p_5) + p_5(1-p_1)(1-p_2) + p_1p_2p_5 =$$
$$= 0.9 \cdot 0.5 \cdot 0.1 + 0.5 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.1 \cdot 0.5 + 0.9 \cdot 0.5 \cdot 0.9 = 0.5.$$

Na sličan način mogu se odrediti i vjerovatnoće za preostale bite parnosti. Naredni problem je kako izvršiti ažuriranje vjerovatnoća pojedinih simbola na osnovu vjerovatnoća koje su dobijene iz bita parnosti. Pogledajmo Sliku VIII.3(b). U prvi čvor pristižu tri vjerovatnoće jedinice u ovom bitu, 0.5, 0.432 i 0.372, zajedno sa procjenom iz kanala 0.9. Treba ih obraditi i ponovo proslijediti prema čvorovima parnosti.

Međutim, sada svakom čvoru parnosti prosljeđujemo različitu vjerovatnoću. Prilikom formiranja te vjerovatnoće ne koristimo vjerovatnoću koju je čvor proslijedio prema c_1, već samo dvije preostale, zajedno sa podatkom dobijenim iz kanala. Označimo tri vjerovatnoće koje se agregiraju u čvoru kao π_1, π_2 i π_3, kao i vjerovatnoću dobijenu iz kanala π_K. Pseudovjerovatnoća jedinice koja se prosljeđuje u prvu granu može se sračunati kao:

$$\pi_2\pi_3\pi_K.$$



Slika VIII.3. (a) Početno stanje za prvi bit poruke; (b) Vjerovatnoće dobijene u prvom koraku iz bitova parnosti za prvi bit; (c) Vjerovatnoće poslate prema bitovima parnosti za prvi bit u drugom koraku algoritma

Kako je riječ o vjerovatnoćama, ovaj postupak bi uvijek vodio ka smanjivanju vjerovatnoća. Stoga, istom logikom možemo dobiti pseudovjerovatnoću nule:

$$(1-\pi_2)(1-\pi_3)(1-\pi_K).$$

Kako suma vjerovatnoća mora biti 1, možemo da uvedemo korektivni faktor k, tako da je:

$$k[\pi_2\pi_3\pi_K+(1-\pi_2)(1-\pi_3)(1-\pi_K)]=1.$$

Dakle, umjesto $\pi_2\pi_3\pi_K$ radimo sa normalizovanom vrijednošću:

$$\frac{\pi_2\pi_3\pi_K}{\pi_2\pi_3\pi_K+(1-\pi_2)(1-\pi_3)(1-\pi_K)}.$$

U našem slučaju dobijamo:

$$\frac{0.436\cdot 0.372\cdot 0.9}{0.436\cdot 0.372\cdot 0.9+(1-0.436)(1-0.372)(1-0.9)}\approx 0.8047$$

$$\frac{\pi_1\pi_3\pi_K}{\pi_1\pi_3\pi_K+(1-\pi_1)(1-\pi_3)(1-\pi_K)}\approx 0.8421$$

$$\frac{\pi_1\pi_2\pi_K}{\pi_1\pi_2\pi_K+(1-\pi_1)(1-\pi_2)(1-\pi_K)}\approx 0.8725.$$

Proces se za sve bite ponavlja kroz više iteracija. Pravilo zaustavljanja algoritma je ili broj iteracija ili situacija kada kroz nekoliko iteracija nema važnije razlike između dobijenih rezultata. U finalnom koraku, vjerovatnoća stanja u posmatranom bitu dobija se na osnovu sve četiri vjerovatnoće: ulazne vjerovatnoće dobijene iz kanala i vjerovatnoća dobijenih u procesu provjera parnosti:

$$\frac{\pi_1\pi_2\pi_3\pi_K}{\pi_1\pi_2\pi_3\pi_K+(1-\pi_1)(1-\pi_2)(1-\pi_3)(1-\pi_K)}.$$

Tipičan broj iteracija je desetak ili nešto više. Odluka se donosi poređenjem dobijenih vjerovatnoća sa pragom koji je uobičajeno postavljen na 0.5. Napominjemo da konstrukcija grafa koji smo mi predstavili nije pogodna za praktične primjene. Naime, ako pogledamo Sliku VIII.4, na njoj smo izdvojili ukupno četiri grane. Ovakva struktura u grafu predstavlja ciklus dužine 4. To nije jedini ciklus ovakve dužine u predmetnom grafu. Postojanje ciklusa ukazuje na povezanost i zavisnost pojedinih odluka donesenih po bitovima, što bi se trebalo izbjeći. Stoga se po pravilu koriste samo one strukture, odnosno kodovi kod kojih se ne pojavljuju ciklusi male dužine.

Slika VIII.4. Tenerov graf sa naglašenim ciklusom dužine četiri

U praksi, sa porastom broja bita parnosti predmetne formule počinju da budu komplikovane za evaluaciju, a greška u operacijama sa pokretnim zarezom može da propagira i da donese grešku u proces dekodiranja. Stoga se obično, umjesto proračuna (pseudo)vjerovatnoća, koriste LLR odnosi, kao kod turbo-kodova.

Prilikom dizajna LDPC kodova obično se prate neke procedure. Ovdje ćemo pokazati bazičnu Galagerovu šemu. Svaki bit učestvuje u J provjera parnosti i svaka parnost provjerava K bita. Pošto je broj jedinica u kontrolnoj matrici isti, bilo da računamo kolone ili vrste, tada važi da je:

$$Jn = K(n-k),$$

gdje je n dužina kodne riječi (odnosno broj kolona matrice), a $n-k$ broj redova kontrolne matrice. Jednostavnim manipulacijama dobijamo da je kodni odnos ovako definisanog LDPC koda:

$$\frac{k}{n} = 1 - \frac{J}{K}.$$

Iz ovakvih kodova mogu se dobiti kodovi izmijenjenih karakteristika, postupcima koje smo pominjali, kao što je „bušenje“ (engl. *puncturing*), odnosno izostavljanje pojedinih nepotrebnih bita u kodnoj riječi. Ako je, na primjer, $J=3$ i $K=4$, dobija se $n-k=3n/4$. Galager je preporučio da se prvih $n/4$ redova kontrolne matrice dobiju tako što se formiraju vrste sa po četiri jedinična bita pomjerena postepeno:

$$\begin{array}{cccccccccccccc}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \uparrow \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \cdots & 0 & 0 & 0 & 0 & n/4 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 & 1 & \downarrow
\end{array}.$$

Narednih $n/4$ redova dobija se permutacijom ove matrice, a, konačno, preostalih $n/4$ redova dobija se drugom permutacijom kolona ove matrice.

Postoje i **neregularni LDPC kodovi**. Kod ovih kodova broj jedinica u redovima i kolonama nije konstantan. Optimalan broj jedinica po redovima i kolonama dobija se posebnom tehnikom. Predmetni kodovi u praksi daju bolje rezultate nego regularni koje smo opisali. Ideja je da se neki bitovi dodatno zaštite provjerama

parnosti i da su neke provjere parnosti pouzdanije kako bi dekoder u procesu dekodiranja (barem u početku) brzo dostizao dekodirajuće vrijednosti.

LDPC kodovi su snažna alternativa turbo-kodovima. U trenutku pisanja ove knjige postoji i dalje značajan prostor za istraživanje: (a) performanse kodova za brojne praktično bitne tipove kanala nisu određene, a samim tim nisu poznate ni optimalne klase kodova koje treba primjenjivati kod tih tipova kanala, (b) optimizacija iregularnih LDPC kodova, posebno za određene tipove kanala, i dalje je problematična, (c) implementacija kodera i dekodera u savremenim hardverskim strukturama i dalje traži poboljšanja, (d) problem patenata kod LDPC kodova (osnovni patenti su bili u vlasništvu *France telecoma*) itd.

VIII.7 Golajevi kodovi

Najpoznatiji nebinarni kod koji je u stanju da ispravi više od jedne pogreške je Golajev ternarni (11,6) kod (dobio ime po *Marselu Golaju*, engl. *Marcel Golay*, švajcarskom matematičaru, fizičaru i pioniru teorije informacija) koji je u stanju da ispravi dvije pogreške. Ovaj kod definisan je generatorskim polinomom $\mathbf{g}(x) = x^5 + x^4 - x^3 + x^2 - 1 = x^5 + x^4 + 2x^3 + x^2 + 2$, na polju u kojem važi da je $x^{11} = 1$. Golaj je ovaj kod otkrio 1949. i do danas je ovo jedini poznati perfektni nebinarni kod koji je u stanju da dekodira više od jedne pogreške. Golaj je inače razvio i binarni perfektni kod (23,11) koji je u stanju da ispravi do tri pogreške. Kontrolna i generatorska matrica Golajevog ternarnog koda koji ispravlja tri pogreške daju se obično u sljedećoj formi:

$$\mathbf{H} = \begin{bmatrix} 1 & 2 & 2 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 2 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 2 & 2 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 2 & 2 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 \end{bmatrix}.$$

Interesantnan je podatak da su kodovi inače bili razvijeni par godina ranije, od strane Virtakalija (fin. *Juhani Virakallio*), za potrebe zapisa rezultata fudbalskih

utakmica. Alternativno Golajevom kodu možemo da koristimo Rid−Solomonove kodove ili varijante BCH kodova prilagođene nebinarnim kodovima. Euklidski algoritam je naš prvi izbor u procesu dekodiranja, ali je potrebno znati da su za ovu namjenu razvijeni i drugi postupci, od kojih su pojedini posebno „skrojeni" za neki od posmatranih kodova, dok drugi imaju opštiju namjenu.

VIII.8 Kodovi u komunikacionim i računarskim standardima

Udžbenik je okrenut korišćenju kodova principijelno, bez ostvrtanja na konkretno okruženje koje postoji u primjeni kodova u informaciono-komunikacionim aplikacijama. Uostalom, to je i uobičajen način na koji se izučava materija u bazičnim kursevima teorije informacija i kodova. Ovdje nam je za cilj da ukažemo da su proučavani kodovi široko raspostranjeni i upotrebljivani u brojnim računarskim i komunikacionim standardima. Naslov ove sekcije je pretenciozan. Stvarno sagledavanje ove problematike bi zahtijevalo mnogo duži tekst od ovog udžbenika, pa smo se opredijelili da predmetnu materiju izložimo kroz nekoliko primjera kako se neki od učenih kodova izvora i kanala koriste u zapisima digitalnih fajlova, odnosno u komunikacijama. U narednim sekcijama dajemo nekoliko informacija vezanih za primjene nekih od učenih algoritama.

VIII.8.1 TIFF format zapisa

TIFF format zapisa (skraćnica od engl. *Tag-based file format*) jedan je od najpoznatijih i najdugovječnijih standarda za zapis digitalne slike. U trenutku pisanja ove knjige aktuelna je šesta revizija ovog formata, koji je još, sada već dosta davne, 1986. godine razvila korporacija Aldus. O ovom formatu zapisa danas uglavnom brine korporacija Adobe. Za sagledavanje složenosti ovog zapisa najbolje je konsultovati Adobeovu dokumentaciju koja je dostupna na internetu. Sam TIFF se sastoji od više nivoa. Osnovni se naziva *baseline* i u njemu postoji mogućnost korišćenja dva algoritma kompresije: PackBits i modifikovanog Hafmenovog algoritma. PackBits algoritam je, suštinski, RLE kodiranje koje se provodi na bajtovima podataka. Modifikovani Hafmenov kod ovdje uključuje RLE kodiranje zajedno sa Hafmenovim kodiranjem. Primjenjuje se samo kod slika koje su crne i bijele (npr. faks dokumenti). U terminologiji TIFF formata ovo se često naziva kompresija drugog nivoa. Hafmenova tabela je kod ovog koda unaprijed poznata (statička). U tabeli se nalaze kombinacije bijelih i crnih polja (na primjer: 10101, što znači: bijelo-crno-bijelo-crno-bijelo).

Drugi nivo u TIFF formatu naziva se *TIFF Extensions* (proširenja). U ovom slučaju postoji više mogućnosti za ostvarivanje kompresije. Prva dva nivoa su CCITT grupa 3 i grupa 4 (ponekad se nazivaju T.4 i T.6 kodiranja). Oba se odnose na prethodno

opisani modifikovani Hafmenov kod, to jest oba se primjenjuju kod dokumenata sa isključivo crnom i bijelom bojom (dakle, potencijalno kod faks dokumenata). Ovdje je bitna razlika u tome (kao što je opisano u Poglavlju III) da se primjenjuje varijanta READ kodiranja, gdje se samo neki od redova kodiraju RLE postupkom, dok se ostali redovi kodiraju u odnosu na ove redove. Postoji nekoliko modova u kodiranju, od kojih je ovdje za nas posebno značajan tzv. vertikalni mod, u kome se zapravo ne vrši kompresija modifikovanim Hafmenovim kodom, već se tekući red poredi sa prethodnim i kodiranje obavlja po šemi u kojoj se odgovarajućim kodom bilježe samo odnosi pozicija gdje se prešlo sa crnog na bijelo ili obrnuto. Na primjer, ako je na istoj poziciji došlo do promjene sa istom bojom, kodira se sa 1, ako je pomjereno za jednu tačku (piksel) udesno − sa 011, piksel ulijevo − 010, dva piksela udesno − 00011, dva piksela ulijevo − 000010 itd.

Pored ovih kodnih postupaka koji se odnose, kao što smo vidjeli, na dokumenta koja isključivo imaju crnu i bijelu boju (binarne slike i faksove), postoje i kodne šeme koje se odnose na slike prikazane sivom skalom i slike u boji. Za nas je nainteresantnija treća kodna šema u proširenjima TIFF standarda, a odnosi se na LZW kodiranje. Napominjemo da četvrti nivo kompresije nije od interesa za ovu knjigu jer se odnosi na kompresiju sa gubicima. Kod primjene LZW algoritma slika se dijeli u linije ili trake koje bi trebale da imaju približno 8KB podataka. Razlog za ovaj izbor leži u težnji da se jedan LZW algoritam obavlja nad količinom podataka iz memorije čak i na malim mašinama, a istovremeno se ovo pokazalo kao količina koja daje dobar stepen kompresije. Veličina LZW tabele je ovdje 12 bita, što znači da se može upisati do 4096 elemenata, s tim da se počinje sa kodom izvora koji tipično ima 256 vrijednosti. Naime, osvjetljaji u slici se tipično kodiraju od 0 (crno) do 255 (bijelo), a na sličan način se kodiraju i osnovne boje u slici. Ako su trake male (kako smo gore preporučili), obično se ne koristi 12 bita, već se tabela ograničava na 11 bita. Naravno, u uslovima kada radimo sa sistemima danas uobičajenih performansi, možemo da uzmemo veće trake i koristimo čitav rječnik. Upisivanje u rječnik ne počinje na poziciji 256, već tek na poziciji 258 jer se pozicije 256 i 257 rezervišu za specijalne karaktere (256 je *Clear* kod za brisanje, dok je 257 *EndOfInformation* kod za kraj poruke). Ostatak procesa kodiranja je u potpunosti isti kao onaj koji smo opisali u Poglavlju III, imajući na umu pomenuto ograničenje veličine rječnika. LZW postupak ne čeka da se ispuni kompletan rječnik, već se na poziciju 4093 ili 4094 šalje *Clear* kod i punjenje rječnika počinje od početka. Da bi se izbjegle eventualne nejasnoće, svaka traka piksela prilikom kodiranja počinje *Clear* simbolom, a završava *EndOfInformation* kodnim simbolom. Da zaključimo, LZW kod koji je implementiran u okviru TIFF standarda praktično je isti kao onaj koji smo učili, uz nekoliko bitnih tehničkih detalja vezanih za implementaciju (veličina rječnika, podjela slike u trake, specijalni simboli itd.).

VIII.8.2 IEEE 802.11

IEEE (engl. *Institute of Electrican and Electronic Engineers*) je najveće međunarodno profesionalno udruženje. Okuplja preko 400.000 specijalista u oblastima elektrotehnike i računarstva iz gotovo svih država na svijetu. Pored ostalih aktivnosti, IEEE je zaslužan za uspostavljanje mnoštva standarda u brojnim oblastima elektrotehnike, elektronike i informatike. Jedan od standarda na koji ćemo se ovdje osvrnuti je IEEE 802.11. Ovaj standard pripada skupu IEEE 802 protokola namijenjenih za LAN mreže (engl. *Local Array Network*), dok se 802.11 odnosi na protokole vezane za bežičnu varijantu mreža. Predmetni standardi obrađuju različita frekvencijska područja i slučajeve korišćenja. Namjena im ide od domova i zgrada, do industrijskih pogona, djelova gradova sa veoma različitim fizičkim konfiguracijama i problemima koji se pojavljuju. Stoga je neophodno da se razviju kvalitetni postupci za korekciju svih tipova grešaka koje se mogu pojaviti u bežičnim komunikacijama, na visokim frekvencijama i sa ogromnom količinom prenesenih bita (ovi standardi danas moraju da omoguće prenos multimedijalnih sadržaja bez uznemiravanja korisnika). Kako IEEE 802.11 obuhvata veliki broj standarda (dvadesetak), teško je obuhvatiti sve tipove kodova za korekciju pogreški koji su se tokom vremena razvijali. Neki do njih ne bi bili ni upotrijebljeni da su znanja koja danas posjedujemo bila raspoloživa u trenutku kada su usvojeni.

Ovdje ćemo obraditi primjenu kodiranja kanala kod IEEE 802.11 a/g standarda koji se primjenjuju za potrebe bežičnih komunikacija WLAN (engl. *Wireless Local Array Network*). Premda ne postoji posebna obaveza da se koristi ovaj kod, uobičajeni izbor za kodiranje kanala kod ovog standarda je konvolucioni kod sa kodnim odnosom $R = 1/2$, sa dva generatorska kodera:

$$\mathbf{g}_0(x) = x^6 + x^4 + x^3 + x + 1$$
$$\mathbf{g}_1(x) = x^6 + x^5 + x^4 + x^3 + 1.$$

Predmetni kod se može transformisati primjenom „bušenja“ u kodove sa većim kodnim odnosom za slučaj kada je stanje u kanalu povoljnije. Kod sa kodnim odnosom $R = 3/4$ dobija se tako što se propuštaju samo biti na parnim pozicijama u kodnoj riječi iz donjeg kodera (ovoga sa polinomom $\mathbf{g}_1(x)$). Kod sa kodnim odnosom $R = 2/3$ dobija se tako što se iz gornjeg kodera eliminiše svaki treći bit, dok se iz donjeg kodera eliminiše bit koji prethodi bitu koji je eliminisan u gornjem koderu (ponovo svaki treći, ali sada drugačije raspoređen). U praktičnoj realizaciji ovog sistema ne vrši se konverzija serije u paralelu, već se povorka bitova prosljeđuje tako što se prethodno dva puta vrši interliving. Blokovi bitova iz drugog interlivera se zatim modulišu posebnim kodnim šemama. Naša knjiga nije vezana za fizički nivo sistema, tako da ćemo taj dio ovdje izostaviti. Samo napominjemo da se fizička modulacija poruke prilikom slanja u kanal obavlja poznatim telekomunikacionim šemema kao što su QAM, BPSK, QPSK itd.

VIII.8.3 Turbo-kodovi kod LTE

LTE (engl. *Long-term evolution*) je telekomunikacioni standard koji se koristi uglavnom kod mobilnih telekomunikacionih mreža četvrte generacije. Ovaj turbo-koder, sa kodnim odnosom $R = 1/3$, već je prikazan na Slici VII.8. Pored sistematskih bita koji se prenose direktno, kodne grane mogu koristiti kodere sa malim brojem ćelija u registru (polinome malog reda):

$$\mathbf{g}(x) = x^3 + x + 1$$

ili

$$\mathbf{g}(x) = x^3 + x^2 + 1.$$

Moguća je kombinacija da gornja grana koristi prvi koder, a donja drugi ili obrnuto (da koderi ne budu isti).

Dizajn interlivera je specifičan. Posmatra se blok od K bita koji su ulazni u interliver $X(k)$, $k = 0,1, \ldots, K-1$. Izlazni biti iz interlivera računaju se kao:

$$X'(k) = X(\mathrm{mod}[f_1 k + f_2 k^2, K]).$$

Parametri f_1 i f_2 biraju se po posebnoj tabeli i zavise od izbora K. Na primjer, za $K = 40$ je $f_1 = 3$ i $f_2 = 10$, dok je za $K = 416$, $f_1 = 25$ i $f_2 = 46$. Dekoder ovog sistema veoma je sličan dekoderu koji je prikazan na Slici VII.14.

Možemo zaključiti da je primjena kodova i za kodiranje izvora i za kodiranje kanala u standardima veoma slična onome što je učeno, uz niz implementacionih praktičnih i tehničkih detalja. Bitno je znati da se kod komunikacionih standarda dobijeni kod prenosi na fizički modulacioni nivo, te da se dobijene kodne riječi moraju pretvoriti u odgovarajući signal, koji se na osnovu modulacione šeme prenosi prema prijemniku. Ipak riječ je o temi koja je vezana za telekomunikacije, odnosno koja izlazi iz oblasti interesovanja našega kursa.

VIII.9 Zadaci i softverska realizacija

VIII.9.1 Riješeni zadaci

8.1. Pokazati da CRC sa generatorskim polinomom $1 + x^2 + x^3 + x^4$ ne uspijeva da detektuje dvije pogreške kada su razdvojene za distancu koja je umnožak broja 7.

Rješenje: Uzmimo da su se pogreške dogodile na razmaku od 7 mjesta, što znači da se sindrom može zapisati kao:

$$\mathbf{s}(x) = (x^k + x^{k+7})\mathbf{d}(x).$$

Stoga je potrebno provjeriti da li je generatorski polinom djelilac polinoma $(1+x^7)$.

$$x^7+1 : x^4+x^3+x^2+1 = x^3+x^2+1$$
$$\underline{x^7+x^6+x^5+x^3}$$
$$\underline{x^6+x^5+x^3+1}$$
$$\underline{x^6+x^5+x^4+x^2}$$
$$\underline{x^4+x^3+x^2+1}$$

Dakle, možemo zaključiti da pogreške neće biti detektovane ako se pojave na razmaku koji je multipl broja 7.

8.2. Pokazati da CRC sa generatorskim polinomom $1+x+x^2+x^3$ ne uspijeva da detektuje dvije pogreške kada su razdvojene za 4 bita.

Rješenje: Postupak je isti kao u prethodnom zadatku. Dvije greške se neće uočiti na rastojanju koje je umnožak četvorke ako je x^4+1 djeljivo sa predmetnim generatorskim polinomom. Dakle, provjeravamo:

$$x^4+1 : x^3+x^2+x+1 = x+1$$
$$\underline{x^4+x^3+x^2+x}$$
$$\underline{x^3+x^2+x+1}.$$

8.3. Neka je $\mathbf{g}(x)=x^3+x+1$ generatorski polinom za CRC. Odrediti „pametan“ algoritam za računanje CRC-a za niz:

$$11100011010100011001111 0,$$

tako da se izbjegne „dosadno“ dijeljenje.

Rješenje: Dijeljenje je operacija koja se svodi na oduzimanje, a oduzimanje se kod binarne algebre svodi na sabiranje po modulu 2, odnosno na ekskluzivno ili operaciju. Procedura koja može biti relativno brza i efikasna u predmetnom slučaju može se sastojati od određivanja jedinice najviše važnosti, a zatim se od te jedinice i naredna 4 bita oduzme kombinacija [1,0,1,1]. Procedura se zatim nastavlja sa jedinicom najveće važnosti u ostatku riječi. Navedeni koraci u predmetnoj poruci su dati:

11100011 01010001 10011110
01010011 01010001 10011110
00001011 01010001 10011110
00000000 01010001 10011110
00000000 00001001 10011110
00000000 00000010 10011110
00000000 00000000 01011110
00000000 00000000 00000110.

Ovdje vidimo da je ostatak dijeljenja 110. Procedura je obavljena relativno brzo. Postoje i drugi postupci za još efikasnije obavljanje predmetne procedure.

8.4. Postoji li binarni kod sa 17 riječi i minimalnom distancom 3, dužine 7?

Rješenje: Kod sa minimalnom distancom 3 može da ispravi jednu riječ. Dužina 7 podrazumijeva da ima ukupno mogućih riječi $2^7 = 128$. Ako želite da postignete da može da ispravlja jednu kodnu riječ, oko svake ispravne kodne riječi mora se formirati sfera koja pokriva, pored ispravne kodne riječi, i susjednih sedam, koje se od ispravne razlikuju na jednom bitu. Stoga, svakoj ispravnoj kodnoj riječi odgovara sfera dimenzije 8, a to dalje znači da možemo imati najviše

$$2^7/8 = 16 \text{ ispravnih kodnih riječi.}$$

Dakle, nije moguće spakovati u dati prostor 17 ispravnih kodnih riječi, odnosno spakovati 17 sfera.

8.5. Postoji li binarni kod sa 29 kodnih riječi, minimalnom distancom 3, dužine 8?

Rješenje: Primjenom ispitivanja pakovanja sfera dobijamo:

$$2^8/9 = 28.44,$$

čime ponovo potvrđujemo da nije moguće imati kod datih karakteristika.

8.6. Postoji li binarni kod sa 7 kodnih riječi, minimalnom distancom 5, dužine 8?

Rješenje: Ponovimo proceduru provjere na osnovu pakovanja sfera. Ukupan broj mogućih kombinacija sa 8 bita je $2^8 = 256$. Za ostvarivanje pakovanja potrebno je u ovom slučaju formirati sferu koja pored ispravne kodne riječi obuhvata i one riječi koje su na rastojanju 1 (ima ih 8) i one koje su na rastojanju 2 od ispravne kodne riječi (ima ih $\binom{8}{2} = 28$). Dakle, ukupan broj sfera koje je teorijski moguće spakovati u dati prostor je:

$$\frac{2^8}{1+8+28} = 6.92.$$

I u ovom slučaju smo prosto, na osnovu pakovanja sfera, zaključili da nije moguć kod sa traženim karakteristikama.

8.7. Postoji li binarni linearni kod dimenzije 2, sa minimalnom distancom 3 i dužinom bloka 5?

Rješenje: Iz postavke zadatka imamo da je $k = 2$, $q = 2$, $d = 3$ i $n = 5$. Po Gilbert–Varšamovoj granici slijedi:

$$q^{n-k}-1>(q-1)\binom{n-1}{1}+\cdots+(q-1)^{d-3}\binom{n-1}{d-3}+(q-1)^{d-2}\binom{n-1}{d-2}$$

$$2^3-1>\binom{4}{1}.$$

Dakle, pošto je navedena nejednakost zadovoljena, moguće je konstruisati ovakav kod.

8.8. Postoji li binarni linearni kod dimenzije 3, sa minimalnom distancom 3, sa blokom dužine 6?

Rješenje: Iz postavke zadatka imamo da je $k=3$, $d=3$, $q=2$ i $n=6$. Po Gilbert–Varšamovoj granici slijedi:

$$2^3-1>\binom{5}{1}.$$

8.9. Postoji li binarni linearni kod dimenzija 3, sa minimalnom distancom 3 i dužinom bloka 5?

Rješenje: Iz postavke zadatka imamo da je $k=3$, $d=3$, $n=5$ i $q=2$. Po Gilbert–Varšamovoj granici slijedi:

$$q^{n-k}-1>(q-1)\binom{n-1}{1}+\cdots+(q-1)^{d-3}\binom{n-1}{d-3}+(q-1)^{d-2}\binom{n-1}{d-2}$$

$$2^3-1>\binom{4}{1}.$$

Dakle, pošto je navedena nejednakost zadovoljena, moguće je konstruisati ovakav kod.

8.10. Postoji li binarni linearni kod dimenzija 3, sa minimalnom distancom 4 i dužinom bloka 8?

Rješenje: Posmatrajmo Gilbert–Varšamovu granicu:

$$2^{8-3}-1>\binom{7}{1}+\binom{7}{2}$$

$$31>7+21.$$

Dakle, Gilbert–Varšamova granica je zadovoljena, pa predmetni kod postoji.

8.11. Postoji li binarni linearni kod dimenzija 2, sa minimalnom distancom 4 i dužinom bloka 7?

Rješenje: Ponovo posmatrajmo Gilbert−Varšamovu granicu:

$$2^5 - 1 > \binom{4}{1} + \binom{4}{2} = 4 + 6.$$

Kako je uslov ove granice zadovoljen, možemo zaključiti da je predmetni kod moguće konstruisati.

8.12. Postoji li binarni kod sa 9 kodnih riječi, minimalnom distancom 3 i sa dužinom 5?

Rješenje: Ne, navedeni kod ne postoji pošto nije zadovoljena Hemingova granica:

$$\frac{2^5}{1+5} \geq 9.$$

8.13. Postoji li binarni kod sa 17 kodnih riječi, sa minimalnom distancom 5 i sa dužinom 8?

Rješenje: Oko svake od 17 kodnih riječi trebalo bi da možemo da kreiramo sferu na distanci 2. Ovo se može provjeriti sljedećom nejednakošću:

$$\frac{2^8}{1+8+\binom{8}{2}} \geq 17$$

koja nije zadovoljena, pa kod sa predmetnim karakteristikama ne može postojati.

8.14. Imamo primljenu sekvencu [0.8 0.2 – 0.3 0.5 0.4 0.1 0.9 – 0.8]. Putem Adamardove matrice dekodirati poruku.

Rješenje: Pomnožimo Adamardovu matricu sa vektorom koji predstavlja primljenu sekvencu:

$$\mathbf{H}_8 r^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0.8 \\ 0.2 \\ -0.3 \\ 0.5 \\ 0.4 \\ 0.1 \\ 0.9 \\ -0.8 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 1.8 \\ 1.2 \\ 0 \\ 0.6 \\ -2.2 \\ 0.4 \\ 2.8 \end{bmatrix}.$$

Najveća apsolutna vrijednost primljenog vektora je na poziciji 8 i iznosi 2.8. Dakle, dekodirana kodna riječ je [1 −1 −1 1 −1 1 1 −1].

8.15. Posmatrajmo sljedeću kodnu riječ: 1 1 0 0 1 1 0 0 1 0 0 1 0 1 0 1. Poruku dekodirati Rid−Mulerovim kodom.

Rješenje: Ako je kodna riječ smještena u vektor **z**, tada možemo izvršiti njeno dekodiranje množenjem $\mathbf{RM}_4$ sa $\mathbf{z}^{\mathrm{T}}$, čime dobijamo:

$$\mathbf{RM}_4\mathbf{z}^T = \begin{bmatrix} 4 \\ 4 \\ 2 \\ 5 \\ 8 \end{bmatrix}.$$

Dakle, peta kolona, koja predstavlja sve jedinice, daje najveći „odziv“, pa zaključujemo da u ovom slučaju predstavlja ispravnu kodnu riječ.

8.16. Dokazati izraz za determinantu Vandermondove matrice i pokazati da navedeni izraz važi za slučaj konačnih polja.

Rješenje: Potrebno je za matricu:

$$\mathbf{VM} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & x_4 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & x_4^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}$$

dokazati da joj je determinanta jednaka

$$\det(\mathbf{VM}) = (-1)^{n(n-1)/2} \prod_{i<j} (x_i - x_j),$$

te da navedena relacija važi za konačna polja $x_i \in \{0, 1, \ldots, P\text{-}1\}$, gdje je P prost broj, dok su sabiranje i oduzimanje definisani po modulu prostog broja P. Dokaz obavimo rekurzivno. Za $n = 2$ dobijamo matricu:

$$\mathbf{VM}_2 = \begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}.$$

Determinanta ove matrice je $\det(\mathbf{VM}_2) = x_2 - x_1$. Ovo je u skladu sa tvrdnjom. Sada pretpostavimo da je navedena tvrdnja tačna za matricu dimenzija n, pa je dokažimo za matricu dimenzija $n+1$. Determinanta dimenzija $n+1$ je jednaka:

$$\mathbf{VM}_{n+1}=\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & \cdots & x_n & x_{n+1} \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & \cdots & x_n^2 & x_{n+1}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & x_4^{n-1} & \cdots & x_n^{n-1} & x_{n+1}^{n-1} \\ x_1^n & x_2^n & x_3^n & x_4^n & \cdots & x_n^n & x_{n+1}^n \end{bmatrix}.$$

Determinantu ove matrice možemo sračunati razvojem po posljednjoj vrsti, kao:

$$\det(\mathbf{VM}_{n+1})=\sum_{i=1}^{n+1}(-1)^{n+i}x_i^{n+1}\det\left(\begin{bmatrix} 1 & \ldots & 1 & 1 & \cdots & 1 \\ x_1 & \cdots & x_{i-1} & x_{i+1} & \cdots & x_n \\ x_1^2 & \cdots & x_{i-1}^2 & x_{i+1}^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & \cdots & x_{i-1}^{n-1} & x_{i+1}^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}\right).$$

Pod pretpostavkom da navedena teorema važi za matrice manjih dimenzija, slijedi:

$$\begin{aligned}\det(\mathbf{VM}_{n+1})&=\sum_{i=1}^{n+1}(-1)^{n+i}x_i^{n+1}(-1)^{n(n-1)/2}\prod_{\substack{l<k\\ l\neq i,k\neq i}}(x_l-x_k)= \\ &=(-1)^{n(n+1)/2}\sum_{i=1}^{n+1}(-1)^i x_i^{n+1}\prod_{\substack{l<k\\ l\neq i,k\neq i}}(x_l-x_k) \\ &=(-1)^{n(n+1)/2}\prod_{\substack{i,j=1\\ i<j}}^{n+1}(x_i-x_j).\end{aligned}$$

8.17. Prikazati matrice RM_1, RM_2, RM_3, RM_4. Odrediti dimenzije i minimalnu distancu koda RM_p u zavisnosti od p.

Rješenje: Ponavljamo postupak iz Poglavlja VIII.3. Prve četiri matrice Rid−Mulerovog koda su:

$$\mathrm{RM}_1=\left[\begin{array}{c|c} 0 & 1 \\ \hline 1 & 1 \end{array}\right] \quad \mathrm{RM}_2=\left[\begin{array}{cc|cc} 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right] \quad \mathrm{RM}_3=\left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$\mathrm{RM}_4 = \left[\begin{array}{cccccccc|cccccccc} 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ \hline 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \end{array}\right].$$

Dužina kodne riječi kod Rid−Mulerovog koda je 2^p, dok je broj informacionih bita $p+1$ (koliko ima vrsta odgovarajuće matrice). Minimalno Hemingovo rastojanje kod ovoga koda je 2^{p-1}.

VIII.9.2 Softverska realizacija

A. CRC kod postoji ugrađen u obliku klase comm.CRCGenerator u novijim verzijama MATLAB-a. Kodni polinom je x^8+x^2+x+1. Iz rezultata se uočava da klasa comm.CRCGenerator i sam generator koda posjeduju čitav niz različitih parametara koje ovdje nećemo razmatrati.

```
crc8=comm.CRCGenerator('Polynomial','x^8+x^2+x+1')
crc8 =
System: comm.CRCGenerator
Properties:
Polynomial:             'x^8+x^2+x+1'
InitialConditions:      0
DirectMethod:           false
ReflectInputBytes:      false
ReflectChecksums:       false
FinalXOR:               0
ChecksumsPerFrame:          1
```

Kodiranje se obavlja, suštinski, upotrebom dobijenog koda kao funkcije:

```
kod=crc8([1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0]')
```

B. Napisati program koji kreira generatorske matrice Rid−Mulerovog koda RM_p za proizvoljno p:

```
RM{1}=[0 1;1 1];
for p=2:P
RM{p}=[zeros(1,size(RM{p-1},2)),ones(1,size(RM{p-1},2));RM{p-1},RM{p-1}];
end
RM{4}
    0    0    0    0    0    0    0    0    1    1    1    1    1    1    1    1
    0    0    0    0    1    1    1    1    0    0    0    0    1    1    1    1
    0    0    1    1    0    0    1    1    0    0    1    1    0    0    1    1
    0    1    0    1    0    1    0    1    0    1    0    1    0    1    0    1
    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
```

C. Uraditi zadatak 8.14 u MATLAB-u. Adamardova matrica se generiše funkcijom hadamard:

```
H8=hadamard(8)
     1     1     1     1     1     1     1     1
     1    -1     1    -1     1    -1     1    -1
     1     1    -1    -1     1     1    -1    -1
     1    -1    -1     1     1    -1    -1     1
     1     1     1     1    -1    -1    -1    -1
     1    -1     1    -1    -1     1    -1     1
     1     1    -1    -1    -1    -1     1     1
     1    -1    -1     1    -1     1     1    -1
```

Ako je primljena poruka data kao u zadatku:

```
x=[0.8 0.2 -0.3 0.5 0.4 0.1 0.9 -0.8];
```

slijedi:

```
s=H8*x';
    1.8000
    1.8000
    1.2000
    0.0000
    0.6000
   -2.2000
    0.4000
    2.8000
```

Pozicija maksimuma (po apsolutnoj vrijednosti) je:

```
[m,p]=max(abs(s));
```

Ako je maksimum pozitivan, dekodirajmo odgovarajuću vrstu Adamardove matrice, a u suprotnom, to je njena negativna vrijednost:

```
if(s(p)>0) rez=H8(p,:); else rez=-H8(p,:);end
rez
     1    -1    -1     1    -1     1     1    -1
```

D. Vandermondova matrica (za realne brojeve) u MATLAB-u se dobija funkcijom vander, kojoj se prosljeđuje vektor:

```
vander([1 2 3 4 5])
     1     1     1     1     1
    16     8     4     2     1
    81    27     9     3     1
   256    64    16     4     1
   625   125    25     5     1
```

Uočite da su koeficijenti matrice dati u neobičnom obliku, kao $\mathbf{v}(i)^{n-j}$, gdje je n dužina vektora argumenta **v**, ali se ovo sa par jednostavnih operacija transformiše u bilo koji pogodan oblik, npr. fliplr(vander([1 2 3 4 5])).

E. Postoji više grupa funkcija kojima se može vršiti Rid−Solomonovo kodiranje i dekodiranje u MATLAB-u. Prvo zadajmo parametre koda:

```
n=7; m=3; k=3;
```

Parametri su redom: dužina kodne riječi, broj informacionih simbola i broj bita kojima se prikazuje svaki od simbola. Poruka u obliku Galoaovog polja može se dobiti naredbom gf:

```
infor= gf([5 2 3; 0 1 7],m)
```

Podrazumijevani kodni polinom je $x^3 + x + 1$, ali se on može mijenjati. Kako smo zadali dvije riječi dužine m, mi zapravo planiramo da kodiramo dvije riječi. Podrazumijevani generatorski polinom se dobija sa:

```
genpoly=rsgenpoly(n,k);
```

Naravno, i on je podložan promjenama. Dobijanje kodne riječi se obavlja funkcijom rsenc:

```
kod=rsenc(infor,n,k,genpoly);
kod = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)
Array elements =
     5     2     3     5     4     4     2
     0     1     7     6     6     0     7
```

Kao što se može vidjeti, pored rezultujuće kodne riječi, dobijamo i podatke o kodu. Generišimo sada grešku:

```
E=zeros(2,7);
E(1,2)=3; E(1,6)=2; E(2,4)=5;
ER=gf(E,m);
```

U prvom redu smo generisali dvije pogreške, a u drugom jednu. Primljena poruka je:

```
Primljeno=ER+kod
Primljeno = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)
Array elements =
     5     1     3     5     4     6     2
     0     1     7     3     6     0     7
```

Konačno dekodiramo:

```
[dec,indik] = rsdec(Primljeno,n,k)
 dec = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)
Array elements =
     5     2     3
     0     1     7
indik =
   2
   1
```

Vidimo da je dekodiranje uspjelo, te da smo dobili polazne poruke, te indik pokazuje da su u prvoj riječi ispravljene dvije, a u drugoj jedna pogreška. Ovo su samo izvodi iz opcija vezanih za Rid−Solomonov kod putem predmetnog skupa funkcija, a postoji mogućnost i rada putem odgovarajuće klase podataka.

F. U novijim verzijama MATLAB-a postoji klasa kojom se definiše LPDC koder comm.LDPCEncoder. Kada se kreira koder ove klase, moguće je korišćenjem ovog kodera, kao klase kod=comm. LDPCEncoder, kodirati poruke kod(x). Naravno, koder (konstruktor klase) ima mnoštvo opcija o kojima preporučujemo da se informišete putem odgovarajućeg help sistema u MATLAB-u. Pored ovoga, postoje na MATLAB Central Exchange sajtu i druge realizacije LDPC kodera i dekodera.

G. U trenutku pisanja ovog udžbenika nije postojala kvalitetna funkcija za kodiranje i dekodiranje Golajevih kodova, ali postoji više programa koji se mogu preuzeti s interneta.

PROJEKTI

PROJEKTI

Činjenica je da teorija, ako nije potkrijepljena praksom, danas ima relativno mali značaj. Stoga smo u dizajniranju kurseva, kojima je ovaj materijal namijenjen, poseban naglasak stavljali na rješavanje određenih praktičnih problema i na upućivanje studenata da sami proširuju saznanja sa predavanja i vježbi. Nekada su ti problemi bili u produbljivanju i provjeri teorijskih saznanja, nekada u praktičnim realizacijama, često su bili van redovnog gradiva, pa su studenti morali da savladaju i svojim kolegama prezentiraju one teme iz ove prebogate oblasti koje zbog vremenskog ograničenja nisu bile obuhvaćene redovnom nastavom. Gotovo uvijek su projekti zahtijevali programsku realizaciju koja nije bila ograničena ni na koji programski jezik. Težnja je bila i da se podigne odgovornost kolega: radovi su se morali predati tačno u određenim terminima, braniti tačno u datom terminu, morali su biti veoma ograničenog obima, prezentirani u kratkom vremenu. Na ovaj način smo forsirali fokusirano izlaganje rezulata, preciznost i ono što se danas vodi pod „mekim vještinama“ (engl. *Soft Skills*), a koje su neophodne na tržištu rada. Veliki broj seminarskih radova je sličan (s određenim modifikacijama), a u nastavku su dati grupisani u određene kategorije. Mišljenja smo da će za svakog čitaoca realizacija ovih projekata donijeti dosta koristi.

Spisak projekata

Projekti su grupisani po redosljedu kako gradivo prirodno slijedi: od teorije informacija, kanala, teorije kodova i partikularnih kodova do dizajniranja interaktivnih alata za ovu oblast.

I. Određivanje entropije

I.1. Uzeti tekst na našem jeziku veličine 100KB. Kao izvor upotrijebiti sajtove sa vijestima na našem jeziku. Izbaciti sve specijalne znake osim jedne bjeline između riječi. Svako slovo azbuke pretvoriti u broj od 1 do 30. Odrediti entropiju ovog sistema pod pretpostavkom da su karakteri međusobno nezavisni. Zatim odrediti entropiju pod pretpostavkom da je u pitanju sistem sa memorijom sa pamćenjem od jednog koraka (Markovljev sistem prvog reda). U jednoj varijanti bjeline posmatrati kao dio koda, dok ih u drugom slučaju posmatrati kao prekid poruke (početak riječi ne zavisi od prethodne riječi). Ponoviti operaciju ako je u pitanju Markovljev sistem 2, 3. i 4. reda. Sumirati rezultate u tabeli. Uporediti rezultate s onima koji su dati u literaturi [2]. Dati i osnovne statističke podatke za tekst, prosječnu dužinu riječi, rečenice, prosječan broj riječi u rečenici itd.

I.2. Uzeti tekst na egleskom jeziku veličine 100KB. Kao izvor upotrijebiti sajtove sa vijestima. Izbaciti sve specijalne znake osim jedne bjeline između riječi. Svako slovo abecede pretvoriti u broj od 1 do 26. Odrediti entropiju ovog sistema pod pretpostavkom da su karakteri međusobno nezavisni. Zatim odrediti entropiju pod pretpostavkom da je u pitanju sistem sa memorijom sa pamćenjem od jednog koraka (Markovljev sistem prvog reda). U jednoj varijanti bjeline posmatrati kao dio koda, dok ih u drugom slučaju posmatrati kao prekid poruke (početak riječi ne zavisi od prethodne riječi). Ponoviti operaciju ako je u pitanju Markovljev sistem drugog, trećeg i četvrtog reda. Sumirati rezultate u tabeli. Uporediti rezultate s onima u literaturi [2] (u pitanju su pretežno podaci vezani za naš jezik). Dati i osnovne statističke podatke za tekst, prosječnu dužinu riječi, rečenice, prosječan broj riječi u rečenici itd.

I.3. U posljednje vrijeme veoma često možete čuti podatke o pravljenju mape genoma čovjeka ili drugih živih bića. Sekvencirane su milijarde nukleotidskih baza u ljudskom genomu. Rukovanje ovako velikom količinom podataka zahtijeva veoma sofisticirane metode obrade podataka. Napomenimo da se svaki nukleotid DNA sastoji od baza: adenozin (A), guanin (G), timin (T) i citozin (C). Napisati program koji učitava sadržaj

jednog niza DNA nukleotida (postoje mnogi javno dostupni podaci, a za početak pretrage pogledati https://www.ncbi.nlm.nih.gov/nucleotide/) i koji određuje entropiju sistema bez memorije. Zatim odrediti entropije sistema, pod pretpostavkom da je sistem Markovljev od prvog do osmog reda. Napomenimo da činjenica da entropija relativno brzo opada sa redom Markovljevog procesa ukazuje na to da je gen nevažan (da je dio koji ne nosi informacije o osobinama) ili da je u pitanju gen koji determiniše neki genetski nedostatak. U slučaju da entropija sporo opada sa redom pretpostavljenog Markovljevog procesa, u pitanju je gen koji determiniše osobine.

I.4. Entropija koju smo uveli nije jedina entropijska funkcija. U teoriji se često koristi i Rényijeva entropija, ali i druge funkcije. Proučiti druge entropijske funkcije i uporediti ih s entropijskom funkcijom koju smo koristili, a zatim tumačiti praktičnost drugih entropijskih funkcija. Realizovati i demonstrirati druge entropijske funkcije.

II. ASCII i Grejov kod

II.1. Kreirati potprogram koji pretvara ASCII kod u binarni zapis. Zatim kreirati potprogram koji binarni zapis pretvara u Grejov kod. Napisati potprogram koji pretvara Grejov kod u binarni zapis. Zatim napisati funkciju koja pretvara binarni zapis u ASCII karakter. Napisati funkciju koja vrši promjenu nekog bita binarnog stringa s određenom vjerovatnoćom. Kreirati program koji, na osnovu prethodnih funkcija, radi sljedeće: učitava niz karaktera realnog teksta napisanog u ASCII kodu i pretvara ga u binarni string, zatim, za vjerovatnoće greške od 0.00001 do 0.15 na jednom bitu (50 različitih vrijednosti u tom intervalu) modifikuje string i na kraju vraća dobijeni string u ASCII kod. Odrediti srednju kvadratnu grešku dobijenog stringa u odnosu na poznati string. Ponoviti istu proceduru ako se binarni string pretvori u Grejov kod i na takav string primjenjuju greške.

III. RLE i READ kod

III.1. Jedno od najinteresantnijih proširenja RLE koda predstavlja READ kod. Ovaj kod se koristi kod faks uređaja, kod kojih su dvije uzastopne linije veoma slične. Napominjemo da se u faks uređajima slika sastoji samo od nula i jedinica (crno i bijelo). Prva linja teksta se kodira RLE kodom, dok se u narednih nekoliko linija kodiraju samo razlike između tekuće i linije koja je kodirana RLE kodom. Program koji pišete treba da se

fokusira na READ kodiranje u idealizovanim uslovima, bez posmatranja konkretne primjene na faks uređaju.

IV. Kodiranje izvora

IV.1. Ilustrujte korišćenje MATLAB-ovog Communications toolboxa za kodiranje izvora.

V. Hafmenov kod

V.1. Uzeti tekst na engleskom jeziku, veličine 100KB. Kao izvor upotrijebiti WWW sajtove sa vijestima. Izbaciti sve specijalne znake, interpunkcijske znake i bjeline, a ostaviti samo slova. Svako slovo kodirati brojnom vrijednošću od 1 do 26. Na osnovu broja pojavljivanja pojedinih karaktera izvršiti određivanja Hafmenovog koda. Ako je za svaki karakter ASCII koda korišćeno 8 bita, procijeniti dobijenu kompresiju. Zatim izdvojiti sekvencu od 100 slova i namjerno kreirati greške na dobijenim bitovima u stringu, koji predstavlja Hafmenov kod, i pogledati i tumačiti dobijene rezultate.

V.2. U posljednje vrijeme veoma često možete čuti podatke o pravljenju mape genoma čovjeka ili drugih živih bića. Rukovanje ovako velikom količinom podataka zahtijeva veoma sofisticirane metode obrade podataka. Napomenimo da se svaki nukleotid DNA sastoji od baza: adenozin (A), guanin (G), timin (T) i citozin (C). Napisati program koji učitava sadržaj jednog niza nukleotida DNA (postoje mnogi javno dostupni podaci, a za početak pretrage pogledati https://www.ncbi.nlm.nih.gov/nucleotide/) i koji vrši Hafmenovo kodiranje ovakve sekvence pod pretpostavkom da je sistem bez memorije ili da je Markovljev proces od prvog do osmog reda. Odrediti srednju dužinu kodne riječi po kodnom simbolu. Ako srednja dužina kodne riječi po kodnom simbolu brzo opada, to znači da posmatrani nukleotid ne nosi informaciju (nije važan u procesu sekvencioniranja) ili da determiniše neki genetski nedostatak. U suprotnom, ako prosječna dužina kodne riječi sporo opada, gen nosi koristan genetski sadržaj, odnosno determiniše neku osobinu.

VI. LZ/LZW kod

VI.1. Kreirati sistem za LZ kodiranje i dekodiranje binarnog alfabeta s varijantama različite dužine rječnika simbola, kao i s mogućnošću izbora početnog stanja u riječniku podataka. Za sve potrebne operacije kreirati

odgovarajuće potprograme. Demonstrirati rad koda na većem broju primjera.

VII. Aritmetički kodovi

VII.1. Kreirati sistem (napisati program) za aritmetičko kodiranje. Pretpostaviti parametre koda kao u primjeru iz Poglavlja III.7. Diskutovati način na koji se može odrediti (fiksirati) dužina kodne riječi prilikom kodiranja.

VIII. Kodiranje izvora kod multimedija

VIII.1. Primjena kodiranja izvora na JPEG (engl. *Joint Photographic Expert Group*) algoritam.

VIII.2. Pronađite nekomprimovane digitalne slike (mogu biti sivoskalirane). Primijeniti TIFF (engl. *Tagged Image File Format*) zapis sa LZW kodom, a zatim komprimovati fajl sa JPEG kompresijom sa različitim nivoima kvaliteta kompresije. Odrediti entropije navedene slike, posmatrajući je kao Markovljev sistem nultog reda i prvog reda. Odrediti granice kompresije na osnovu Markovljevih sistema, pa porediti i tumačiti dobijene rezultate.

VIII.3. Primjena LZW kodiranja kod digitalne slike. Opis kompletnog algoritma sa programskom realizacijom.

VIII.4. Primjena kodova za kodiranje izvora kod kodiranja digitalne slike.

IX. Primjena teorije informacija u genetskim/molekularnim istraživanjima

IX.1. Teorija informacija i kodova se sve više koristi u biološkim, molekularnim i genetičkim istraživanjima. U prilogu je dat jedan rad na ovu temu, pa ga detaljno izanalizirajte i realizujte što je više moguće predmetnih procedura. Rad ne treba prevoditi niti prepisivati, već obraditi i prezentirati samo onaj dio koji je razumljiv i koji možete prenijeti ostalim kolegama.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3220916/
Po potrebi, naći i obraditi i drugu literaturu iz ove oblati.
Napomena. Ova tema je već pomenuta u nekoliko projekata (I.3 i V.2).

X. Modeli i simuliranje komunikacionog kanala

X.1. U praksi se rijetko kreirani sistem za prenos informacija testira prvo *in vitro*, nakon što se sistem napravi, već mu se prve provjere obavljaju na simulacionim testovima. Veliki problem predstavlja dobar sistem koji će simulirati na kvalitetan način komunikacioni kanal. Najjednostavniji način (koji treba da realizujete) je da se s određenom vjerovatnoćom generiše pogreška na međusobno nezavisnim pozicijama. Pored ovoga, razrađen je veći broj sistema koji generišu korelirane pogreške. Vaš zadatak je da realizujete sisteme (funkcije) za generisanje pogreški u binarnom kanalu, na osnovu Gilbertovog i Smit−Boven−Džojsovog modela.

X.2. U praksi se rijetko kreirani sistem za prenos informacija testira prvo *in vitro*, nakon što se sistem napravi, već mu se prve provjere obavljaju na simulacionim testovima. Veliki problem predstavlja dobar sistem koji će simulirati na kvalitetan način komunikacioni kanal. Najjednostavniji način (koji treba da realizujete) je da se sa određenom vjerovatnoćom generiše pogreška na međusobno nezavisnim pozicijama. Pored ovoga, razrađen je veći broj sistema koji generišu korelirane pogreške. Vaš zadatak je da realizujete sisteme (funkcije) za generisanje pogreški u kanalu, na osnovu Fričman−Svobodinog i Milerovog modela.

X.3. U teoriji smo objasnili kako se određuje kapacitet kanala bez memorije. Vaš zadatak je da teorijski i praktično opišete način na koji se može odrediti kapacitet kanala sa memorijom, te način na koji se memorija kod kanala može opisati.

X.4. Pretpostavite da raspolažete kanalom koji prenosi binarni signal superponiran Gausovom smetnjom. Napisati program koji eksperimentalno određuje kapacitet kanala u zavisnosti od varijanse šuma. Ponoviti proceduru ako je superponirani šum sa Laplasovom karakteristikom sa različitim parametrima.

XI. Kodiranje kanala

XI.1. Ilustrujte korišćenje MATLAB-ovog Communications toolboxa za kodiranje i dekodiranje kanala, odnosno za kodove koji koriguju pogreške (engl. *error-corecting* kodove).

XII. ARQ saobraćaj

XII.1. Napisati program koji vrši simulaciju pomorskog ARQ saobraćaja. Ovaj tip saobraćaja koristi se kod pomorskih radio-teleks sistema. Koristi se međunarodna kodna azbuka broj dva, kojom se može putem pet bita kodirati 32 karaktera. Svaki od karaktera se u modemu teleprintera mijenja sa sedam bita, od kojih je četiri jedinice i tri nule. Paketski se šalju tri karaktera, a prijemni modem broji jedinice i nule. Nakon uspješnog prijema poruke, prijemni modem zatraži slanje novog paketa, a ako nije primio dobru poruku, zahtijeva ponavljanje iste. Odrediti vjerovatnoću greške da prijemni modem ne može da detektuje grešku u komunikaciji u zavisnosti od vjerovatnoće greške na jednom bitu (pretpostaviti da su greške na bitima nezavisni događaji).

XIII. Hemingov kod

XIII.1. Hemingovi kodovi. Opšta teorija sa realizacijom i binarnih i nebinarnih Hemingovih kodova.

XIII.2. Napisati funkcije koje realizuju Hemingovo kodiranje i dekodiranje binarnog stringa. Funkcije treba da imaju mogućnost rada s ispravljanjem jedne greške i s ispravljanjem jedne greške i detekcijom dvije greške. Funkcije treba da imaju prilagodljivu dužinu kodne riječi, koja se zadaje kao argument. Treba ispitati da li kod radi dobro u uslovima pojave grešaka, a za različite dužine kodne riječi i različite vjerovatnoće pogreške na jednom bitu (posmatrati kao nezavisne veličine) treba nacrtati vjerovatnoću da nema greške u sistemu, da kod ispravi grešku, da kod detektuje pojavu dvije greške i da kod ne može da otkrije grešku.

XIII.3. Kreirati sistem za prenos binarnih informacija sa Hemingovim kodovima (7,4), (8,4), (15,11) i (16,11). Poruka treba da bude zahvaćena Gausovim bijelim šumom sa različitim varijansama (na primjer, u granicama od 0.0001 do 10, sa stotinjak različitih vrijednosti u tom intervalu). Na prijemnoj strani, simbol koji je primljen preko praga 1/2 tumači se kao 1, dok se simbol koji je ispod praga tumači kao 0. Potrebno je, putem Monte Karlo (engl. *Monte Carlo*) simulacije, odrediti vjerovatnoću greške, vjerovatnoću korekcije greške i vjerovatnoću detekcije grešaka u zavisnosti od varijanse šuma.

XIII.4. Kreirati sistem za prenos binarnih informacija kodiranih Hemingovim kodovima (7,4), (8,4), (15,11) i (16,11). Jedan bit se šalje u obliku pet

odbiraka. Poruka treba da bude zahvaćena Gausovim bijelim šumom sa različitim varijansama (na primjer, u granicama od 0.0001 do 10, sa stotinjak različitih vrijednosti u tom intervalu). Na prijemnoj strani, simbol koji je primljen preko praga 1/2 tumači se kao 1, dok se simbol koji je ispod praga tumači kao 0. Prilikom dekodiranja, ako se tri ili više odbiraka primi preko 1/2, podrazumijeva se da je primljena jedinica, dok se u suprotnom podrazumijeva da je primljena 0. Potrebno je, putem Monte Karlo simulacije, odrediti vjerovatnoću greške, vjerovatnoću korekcije greške i vjerovatnoću detekcije grešaka u zavisnosti od varijanse šuma.

XIII.5. Kreirati sistem za prenos binarnih informacija kodiranih Hemingovim kodovima (7,4), (8,4), (15,11) i (16,11). Jedinica se kodira kao sinusoida sa frekvencijom od 10Hz, dok se nula kodira kao sinusoida sa frekvencijom od 0Hz. Poruka treba da bude zahvaćena Gausovim bijelim šumom sa različitim varijansama (na primjer, u granicama od 0.0001 do 10, sa stotinjak različitih vrijednosti u tom intervalu). Na prijemnoj strani mjeri se frekvencija primljene sinusoide i ako je primljena frekvencija od 5Hz i više, podrazumijeva se da je u pitanju jedinica, a u suprotnom da je u pitanju nula. Potrebno je, putem Monte Karlo simulacije, odrediti vjerovatnoću greške, vjerovatnoću korekcije greške i vjerovatnoću detekcije grešaka u zavisnosti od varijanse šuma.

XIII.6. Napisati funkcije koje realizuju Hemingovo kodiranje i dekodiranje binarnog stringa. Npr. funkciju koja dijeli binarni string u podstringove, funkciju koja podstring kodira Hemingovim kodom, funkciju koja dobijenu poruku u Hemingovom kodu vraća u standardni binarni zapis itd. Neka sistem ispravlja jednu grešku, ali neka postoji mogućnost da su u pitanju različite dužine stringova koji se kodiraju. Ova dužina treba da bude parametar funkcije ili je funkcija sama može procijeniti na osnovu zadatog podatka koji je ulazni argument.

XIV. Interliver

XIV.1. Gotovo svi kodovi koje smo uveli podrazumijevaju da se greške pojavljuju u sistemu relativno rijetko i da su međusobno nezavisne. Međusobna nezavisnost grešaka u realnim kanalima je rijedak slučaj. Naime, gotovo svi kanali od praktične važnosti su takvi da ako se dogodi greška u nekom trenutku (na nekom bitu), velika je vjerovatnoća da će se greška događati i u susjednim. Jedan od načina da se ovaj problem ispravi je preko sistema sa interliverom/deinterliverom. Vaš zadatak je da kreirate funkcije koje vrše interliving i deinterliving. Ispravljanje pogreški (do

jedne) u pojedinim kodnim riječima obavlja se preko odgovarajućeg Hemingovog koda. Broj informacionih bita u jednoj kodnoj riječi nakon interlivinga, kao i broj riječi koje se simultano dovode na ulaz interlivera su parametri funkcija. Prikažite rezultate za nekoliko karakterističnih primjera. Kreirane funkcije treba da posjeduju više mogućnosti vezanih za permutacije koje interliver unosi u sistem.

XV. BCH kod

XV.1. Realizovati funkcije koje vrše kodiranje i dekodiranje BCH(15,3) koda. Provjeriti da li funkcije uspješno rade.

XV.2. BCH kodiranje u MATLAB-u korišćenjem MATLAB-ovog Communications toolboxa.

XV.3. Vaš zadatak je da kreirate sistem za prenos binarnih informacija kodiranih BCH kodovima (15,2) i (15,3). Poruka treba da bude zahvaćena Gausovim bijelim šumom sa različitim varijansama (na primjer, u granicama od 0.0001 do 10, sa stotinjak različitih vrijednosti u tom intervalu). Na prijemnoj strani, primljena vrijednost preko praga 1/2 tumači se kao 1, dok se primljena vrijednost ispod praga tumači kao 0. Potrebno je da, putem Monte Karlo (engl. *Monte Carlo*) simulacije, odredite vjerovatnoću greške, vjerovatnoću korekcije greške i vjerovatnoću detekcije grešaka u zavisnosti od varijanse šuma.

XV.4. Vaš zadatak je da kreirate sistem za prenos binarnih informacija kodiranih BCH kodovima (15,2) i (15,3). Jedan bit se šalje u obliku 5 odbiraka. Poruka treba da bude zahvaćena Gausovim bijelim šumom sa različitim varijansama (na primjer, u granicama od 0.0001 do 10, sa stotinjak različitih vrijednosti u tom intervalu). Na prijemnoj strani, vrijednost preko praga 1/2 tumači se kao 1, dok se vrijednost ispod praga tumači kao 0. Prilikom dekodiranja, ako se 3 ili više odbiraka primi preko praga, podrazumijeva se da je primljena jedinica, dok se u suprotnom podrazumijeva da je primljena nula. Potrebno je da, putem Monte Karlo simulacije, odredite vjerovatnoću greške, vjerovatnoću korekcije greške i vjerovatnoću detekcije grešaka u zavisnosti od varijanse šuma.

XV.5. Vaš zadatak je da kreirate sistem za prenos binarnih informacija kodiranih BCH kodovima (15,2) i (15,3). Jedinica se kodira kao sinusoida sa frekvencijom od 10Hz, dok se nula kodira kao sinusoida sa frekvencijom od 0Hz. Poruka treba da bude zahvaćena Gausovim bijelim šumom sa

različitim varijansama (na primjer, u granicama od 0.0001 do 10, sa stotinjak različitih vrijednosti u tom intervalu). Na prijemnoj strani mjeri se frekvencija primljene sinusoide i ako je primljena frekvencija od 5Hz i više, podrazumijeva se da je u pitanju jedinica, a u suprotnom da je u pitanju nula. Potrebno je da, putem Monte Karlo simulacije, odredite vjerovatnoću greške, vjerovatnoću korekcije greške i vjerovatnoću detekcije grešaka u zavisnosti od varijanse šuma.

XV.6. Na kraju BCH dekodirajućeg algoritma javlja se jednačina na osnovu koje treba da se odredi pozicija pogreški. Vaš zadatak je da proučite BCH algoritam, a posebno ovu jednačinu, pa da predstavite najjednostavniji način za rješavanje ove jednačine u opštem slučaju.

XVI. Hardver za kodiranje/dekodiranje kanala

XVI.1. Hardver za dekodiranje Hemingovih i BCH kodova. Na časovima smo se uglavnom upoznali sa sistemima za kodiranje, a vaš je zadatak da pronađete i proučite hardverske sisteme za dekodiranje ovih kodova.

XVII. Teorijski aspekti kodiranja kanala

XVII.1. Koncepti dualnosti kod kodova. Veze ostvarene preko dualnosti kod kodova koje smo proučavali.

XVII.2. Princip proširivanja kodova. Veze ostvarene preko proširivanja kod kodova koje smo proučavali.

XVII.3. Najvažnije asimptotske granice koje se mogu uspostaviti u pogledu određivanja dužine kodne riječi, broja informacionih bita i minimalnog Hemingovog rastojanja. Sastavni dio seminara treba da bude izrada zadataka za vježbanje sa kraja IX poglavlja knjige „Introduction to coding theory“, [5].

XVII.4. Algebarska teorija grupa – detaljnije nego je predstavljeno u udžbeniku. Obavezan dio seminarskog je izrada zadataka iz osmog poglavlja knjige „Mathematics of coding theory“, [4].

XVII.5. Algebarska teorija prstena i polja grupa. Obavezan dio seminarskog je izrada zadataka iz devetog poglavlja knjige „Mathematics of coding theory“, [4].

XVII.6. Provjeriti sve učene kodove za kodiranje kanala, u smislu dostizanja granica II Šenonove teoreme, na seriji primjera i programa, a sa promjenljivim vjerovatnoćama greške po bitu.

XVII.7. Uočili smo da je veoma bitno da odredimo proste polinome. Pretraga za (dobrim) prostim polinomima je dosta složena procedura, koja u najgorem slučaju podrazumijeva dijeljenje sa svim polinomima manjeg stepena. Ako je stepen polinoma veliki, ovo je neprihvatljiva strategija. Stoga su tokom vremena razvijeni brojni postupci kojima je pretraga za prostim polinomima pojednostavljena. Vaš zadatak je da te tehnike pronađete i sublimirate.

XVIII. Konvolucioni kodovi

XVIII.1. Vaš zadatak je da proučite dostupnu literaturu o konvolucionim kodovima, načinu njihovog generisanja (kodiranja) i algoritmima dekodiranja. Poželjna je programska realizacija algoritama kodiranja i dekodiranja konvolucionih kodova.

XIX. Turbo-kodovi

XIX.1. Turbo-kodovi. Ova grupa kodova je jedan od najsavremenijih pristupa u rješavanju problema približavanja Šenonovoj granici za kodiranje kanala. Vaš zadatak je programska realizacija turbo-kodova.

XIX.2. Realizacija soft-input soft-output Viterbijevog algoritma i njegova primjena kod turbo-kodova.

XIX.3. Turbo-kodovi. Ova grupa kodova je jedan od najsavremenijih pristupa u rješavanju problema približavanja Šenonovoj granici za kodiranje kanala. Vaš zadatak je teorijski opis i dokaz razloga za kvalitetan rad ovih kodova.

XX. Golajevi kodovi

XX.1. Binarni Golajevi kodovi, istorija, karakteristika i realizacija kodera i dekodera. U realizaciji se držati principa algebarske kodne teorije i polinomijalnog pristupa.

XXI. Viterbijev algoritam

XXI.1. Upoznajte se sa važnošću i upotrebom Viterbijevog algoritma za konvoluciono dekodiranje. Napišite program i isprobajte funkcionisanje Viterbijevog algoritma za neki poznati kodni sistem (sistem može biti i sa konstantnom dužinom kodne riječi, uključujući Hemingove kodove za ispravljanje jedne greške).

XXII. Rid–Solomonovi kodovi

XXII.1. Realizacija kodera i dekodera kod Rid−Solomonovih kodova.

XXII.2. Rid−Solomonovi kodovi su vjerovatno najpoznatija klasa nebinarnih kodova koji se koriste za ispravljanje više od jedne greške. Opišite ih, povežite sa BCH kodovima, prikažite kodiranje i dekodiranje, te programsku realizaciju. Posebno se fokusirati na sličnosti i razlike u procesu dekodiranja sa BCH kodovima.

XXIII. Rid–Mulerovi kodovi

XXIII.1. Realizacija Rid−Mulerovih kodova prvog reda i veza sa Adamardovom transformacijom.

XXIV. Gopa kodovi

XXIV.1. Gopa kodovi su poznata klasa kodova, definisana putem algebarskih krivih. Vaš zadatak je opis ovih kodova, uključujući algoritme kodiranja i dekodiranja, opis prednosti i mana kodova, te programska realizacija ovih kodova. Poseban naglasak je na karakteristikama ovih kodova i granicama kodiranja koje se mogu postići putem ovih kodova.

XXV. CRC kodovi

XXV.1. Ciklični kodovi i osnovni principi za njihovu realizaciju (hardverski i softverski). Treba realizovati sve softverske realizacije koje možete da postignete na osnovu raspoložive literature.

XXV.2. Opisati CRC postupak, najpoznatije CRC kodove, realizaciju sistema i mjesto CRC postupka u komunikacionim i standardima za zapis podataka u memoriji.

XXVI. LDPC kodovi

XXVI.1. U posljednje vrijeme, iz više razloga, a posebno zbog implementacije navedenih kodova, posebno su popularni engl. *Low-density parity-check* – LDPC kodovi. Vaš zadatak je da napišete seminarski rad na ovu temu, uključujući programsku realizaciju implementacije navedenih kodova.

XXVI.2. Kodovi sa provjerom parnosti male gustine (LDPC) su se posljednjih godina pojavili kao alternative turbo-kodovima i drugim kodovima za kodiranje kanala. Vaš izvještaj treba da sadrži teorijske osnove ovih kodova, da objasni njihove prednosti, prikaže koder i dekoder, te da predstavi programsku realizaciju sistema.

XXVII. Kodiranje i modulacije

XXVII.1. Diferencijalna impulsna kodna modulacija (DPCM) i njena veza s teorijom informacija i kodova. Prediktivno kodiranje.

XXVII.2. Primjena kodova za kodiranje kanala i kodiranje izvora u kombinaciji sa OFDM komunikacionim šemama.

XXVIII. Kodiranje kanala kod multimedijalnih sistema

XXVIII.1. Postoji više savremenih formata zapisa audio-podataka, od kojih je jedan AAC. U ovom standardu kompresije audio-podataka koristi se, između ostalog, i kodiranje sa i bez gubitaka. Razmotriti osnovne elemente kodne teorije (kodove sa i bez gubitaka koji se u ovom komunikacionom standardu koriste). Realizovati ključne rutine u tom procesu.

XXIX. Kodiranje i komunikacioni standardi

XXIX.1. Primjena kodova za korekciju greške u IEEE 802.11 familiji komunikacionih standarda.

XXIX.2. Primjena kodova za korekciju greške u digitalnoj televiziji.

XXIX.3. Primjena kodova za kodiranje kanala kod UMTS komunikacionog standarda.

XXX. Kriptografija

XXX.1. Teorija informacija i kodova, u širem smislu, obuhvata i oblast prikrivenog prenosa informacija, koja se naziva kriptografija. Grubo govoreći, cilj kriptografije je prenos informacija na taj način da samo korisnik kome je ta informacija namijenjena može da je i razumije. Tokom ljudske istorije, prije svega, u vojne svrhe, razvijen je ekstremno veliki broj kriptografskih tehnika. Kreirati i realizovati neke od poznatih (ove šeme se uglavnom navode iz istorijskih, a ne iz praktičnih razloga) sistema, kao što su Cezarova šifra, Vižnerova šifra i Vernanova šifra. Objasniti princip dekodiranja, način formiranja ključa i neke od principa za „razbijanje šifre".

XXX.2. Teorija informacija i kodova, u širem smislu, obuhvata i oblast prikrivenog prenosa informacija, koja se naziva kriptografija. Grubo govoreći, cilj kriptografije je prenos informacija na taj način da samo korisnik kome je ta informacija namijenjena može da je i razumije. Tokom ljudske istorije, u razne svrhe, razvijen je ekstremno veliki broj kriptografskih tehnika. Prvi sistematski sistem za kriptovanje je Digital Encryption Standard (DES). Realizujte sistem za DES. Parametri sistema koji se mogu korisnički definisati, kao što je funkcija f(), ostavljaju vam se na slobodan izbor. U slučaju da je realizacija ovog sistema preambiciozna za vas, probajte da realizujete pojedine blokove (blokove za difuziju i konfuziju). Kako DES ima više arhitektura, kao što su ECB i CBC, opredijeliti se za onu koja vam se čini jednostavnijom.

XXX.3. Kriptovalute uzimaju sve više maha. Vaš zadatak je analiza i opis kriptografskih i elemenata teorije informacija i kodova u *blockchain* trgovini i kod kriptografskih valuta kao što je *bitcoin*.

XXXI. Interaktivno-edukativni projekat

XXXI.1. Vaš zadatak je kreativan i ostavljena vam je velika sloboda. Cilj je kreirati interaktivno sredstvo koje će studenti moći koristiti za samoedukaciju iz teorije informacija i kodova.

XXXII. Spektralne karakteristike slučajnih procesa

XXXII.1. Pronađite više podataka o spektralnim karakteristikama slučajnih procesa i napišite esej/seminarski rad o tome.

SAŽETAK

Udžbenik „Teorija informacija i kodova“ namijenjen je studentima koji prate istoimene kurseve na Elektrotehničkom fakultetu Univerziteta Crne Gore, i na akademskim i na primijenjenim studijama. Dodatno, može se koristiti na postdiplomskim studijama.

Materijal je organizovan u osam poglavlja, koja se mogu grupisati u pet djelova:

- neophodna matematička predznanja − u Poglavlju I;
- teorija informacija − u Poglavljima II (entropija) i III (kompresija);
- komunikacioni kanal − u Poglavlju IV;
- teorija kodova − u Poglavljima V (osnovni pojmovi), VI (blok kodovi) i VII (konvolucioni i turbo-kodovi);
- napredna pitanja u Poglavlju VIII.

Svim poglavljima su pridruženi riješeni problemi i softverske realizacije razmatranih koncepata. Lista projekata za samostalni rad data je na kraju knjige.

ABSTRACT

Textbook „Theory of information and codding“ is intended for students following courses of the same name a the Faculty of Electrical Engineering both for academic and applied studies. In addition, the textbook can be used for the advanced version of the course on graduate studies.

The textbook is organized into eight chapters that can be grouped into five parts:

- the mathematical background is given in Chapter I;
- information theory is given in Chapters II (entropy) and III (compression);
- communication channel in Chapter IV;
- codding theory in Chapters V (basic issues), VI (block codes), and VII (convolution and turbo codes);
- advanced topics in Chapter VIII.

All chapters are accompanied by solved problems and software realizations of considered concepts. A list of projects for self- exercises is given at the end of the textbook.

LITERATURA

[1] R. W. Hamming, Coding and information theory, Prentice-Hall, 1986.

[2] P. N. Ivaniš, D. Drajić, Uvod u teoriju informacija i kodovanje, Akademska misao, Beograd, 2009.

[3] V. Sinković, Informacija, simbolika, semantika, Školska knjiga, 1997.

[4] P. B. Garrett, The mathematics of coding theory, Pearson, 2003.

[5] R. Roth, Introduction to coding theory, Cambridge University Press, 2006.

[6] J. I. Hall, Notes on coding theory, Department of Mathematics, Michigan State University, 2003.

[7] G. Jovanović-Doleček, Teorija vjerovatnoće, Sarajevo, 1990.

[8] S. A. Tretter, Predavanja i drugi materijali iz teorije kodova, Department of Electrical and Computer Engineering, University of Maryland.

[9] D. J. C. MacKay, Information theory, inference and learning algorithms, Cambridge University Press, 2003.

[10] T. Moon, Information theory, Utah State University, predavanja i ostali materijali.

[11] A. Gersho, R. Gray, Vector quantization and signal compression, Kluwer Academic Publishers, 2003 (deveto izdanje), dostupno i putem Google booksa.

[12] Turbo code primer, dostupno onlajn: vashe.org/turbo/turbo_primer_0.0.pdf

[13] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error-correcting coding and decoding: turbo-codes, in Proc. of IEEE Int. Conf. on Communications, May 1993, Ženeva, Švajcarska, 1064−1070.

[14] J. B. Anderson, S. Mohan, Source and channel coding, Kluwer Academic Publishers, 1991.

[15] J. B. Anderson, A. Svensson, Coded modulation systems, Kluwer Academic/Plenum Publishers, 2003.

[16] B. Sklar, A primer on turbo code concepts, IEEE Communications Magazine, Dec. 1997, 94−102.

[17] M. C. Valenti and J. Sun, Turbo codes, DOWLA: Handbook of RF and wireless technologies, 375−400.

[18] Matlab Central turbo-kodovi, http://www.mathworks.com/matlabcentral/fileexchange/39423-turbo-code/content/turbo.m, pristupljeno 29. 12. 2013.

[19] http://www.mathworks.com/matlabcentral/fileexchange/authors/13257, LZW kodovi.

[20] B. Ryabko, A. Fionov, Basics of contemporary cryptography for IT practioners, World Scientific, 2005.

[21] J. Bierbrauer, Introduction to coding theory, Champan & Hall/CRC, 2005.

[22] B. Arazi, A common sense approach to the theory of error correcting codes, MIT Press, 1988.

[23] E. R. Berlekamp, Algebraic coding theory, McGraw-Hill, New York, 1968.

[24] E. R. Berlekamp, editor, Key papers in the development of coding theory, IEEE Press, 1974.

[25] E. Biglieri, P. Divsalar, P. J. McLane, M. K. Simon, Introduction to trellis-coded modulation with applications, Macmillan, 1991.

[26] R. E. Blahut, Theory and practice of error control codes, Addison-Wesley, 1983.

[27] R. E. Blahut, Algebraic codes for data transmission, Cambridge University Press, 2003.

[28] I. F. Blake, R. C. Mullin, The mathematical theory of coding, Academic Press, New York, 1975.

[29] G. C. Clark, J. B. Cain, Error-correction coding for digital communications, Plenum Press, 1981.

[30] J. L. Fan, Contrained coding and soft iterative decoding, Kluwer Academic Publisher, 2001.

[31] C. Nuttelman, Introduction to Monte Carlo simulation, onlajn kurs, dostupno onlajn:

https://www.coursera.org/lecture/excel-vba-for-creative-problem-solving-part-3-projects/introduction-to-monte-carlo-simulation-iozbm

[32] B. D. Fritchman, A binary channel characterization using partitioned Markov chains, IEEE Trans. Inf. Theory, Vol. 13, no. 2, pp. 221–227, Apr. 1967.

[33] E. N. Gilbert, Capacity of a burst-noise channel, Bell Syst. Technol. J., Vol. 39, pp. 1253–1265, Sept. 1960.

[34] Z. Mousavian, K. Kavousi, A. Masoudi-Nejad, Information theory in systems biology. Part I: Gene regulatory and metabolic networks, Semin. Cell. Dev. Biol., Vol. 51, Mar. 2016, pp. 3−16, DOI: 10.1016/j.semcdb.2015.12.007.

[35] T. D. Schneider, A brief review of molecular information theory, Nano Commun. Netw., Vol. 1, No. 3, Sept. 2010, pp. 173−180.

[36] T.6: Facsimile coding schemes and coding control functions for Group 4 facsimile apparatus, ITU-T. November 1988.

[37] J. Li, S. Lin, K. A. Ghaffar, W. E. Ryan, D. E. Costello Jr., LDPC code designs, constructions and unification, Cambidge University Press, 2017.

[38] J. K. Wolf, An introduction to error correcting codes Part 3 – Introduction to LDPC codes, dostupno onlajn:

http://circuit.ucsd.edu/~yhk/ece154c-spr16/pdfs/ErrorCorrectionIII.pdf /

[39] S. B. Wicker, V. K. Bhargava, ed., Reed-Solomon codes and their application, IEEE Press, 1994.

[40] M. Tomlinson, C. J. Tjhai, M. A. Ambroze, M. Ahmed, M. Jibril, Reed-Solomon codes and binary transmission, in: Error-Correction Coding and Decoding. Signals and Communication Technology. Springer, Cham, 2017, pp. 167–179.

[41] E. Weiss, Generalized Reed-Muller codes, Information and Control, Vol. 5, No. 3, pp. 213–222, 1962.

[42] J. Moore, Constant-ratio code and automatic-RQ on transoceanic HF radio services, IRE Transactions on Communications, Vol. 8, No. 1, Mar. 1960, pp. 72–75.

[43] M. Gottscho, C. Schoeny, L. Dolecek, P. Gupta, Software-defined error-correcting codes, in Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), June-July 2016, DOI: 10.1109/DSN-W.2016.67.

[44] A. Shokrollahi, LDPC Codes: An introduction, dostupno onlajn na:

https://www.ics.uci.edu/~welling/teaching/ICS279/LPCD.pdf

[45] V. Guruswami, Bounds on Codes, lekcije za kurs Error-Correcting Codes, 2006, dostupno onlajn na:

https://courses.cs.washington.edu/courses/cse533/06au/lecnotes/lecture5.pdf

[46] B. Z. Shen, A Justesen construction of binary concatenated codes that asymptotically meet the Zyablov bound for low rate, IEEE Transactions on Information Theory, Vol. 39, No. 1, Jan. 1993, pp. 239−242.

[47] J. Waters, QR codes for dummies, John Wiley & Sons, 2012.

[48] Working group on WLAN standards, IEEE 802.11TM Wireless local area networks, oficijelni veb-sajt Radne grupe za standardizaciju WLAN, dostupan onlajn:

http://www.ieee802.org/11/

[49] C. L. Chen, R. A. Rutledge, Error correcting codes for satellite communication channels, IBM Journal of Research and Development, Vol. 20, No. 2, Mar. 1976, pp. 168−175.

[50] C. Paar, J. Pelzl, Understanding cryptography, Springer, 2010.

[51] S. Yang, The capacity of communication channels with memory, PhD thesis, Harward University, 2004, dostupno onlajn na:

http://citeseerx.ist.psu.edu/viewdoc/download?-doi=10.1.1.195.6132&rep=rep1&type=pdf

[52] Э. Л. Блох, О. В. Попов, В. Я. Турин, Модели источника ошибок в каналах передачи цифровой информации, Связь, 1971.

[53] M. Milosavljević, S. Adamović, Osnovi teorije informacija i kodovanje, Univerzitet Singidunum Beograd, ISBN 978-86-7912-610-8, 2017, dostupno na

https://singipedia.singidunum.ac.rs/izdanje/40723-osnove-teorije-informacija-i-kodovanje

POJMOVI, OZNAKE I SKRAĆENICE

POJMOVI [1]

Podatak — Predstava neke pojave, vrijednosti, koncepta.

Informacija — Informacija u savremenom svijetu ima mnogo značenja, ali nema preciznu definiciju. Najpreciznije, informacija je dio poruke koju primamo, a koji možemo na nedvosmislen način razumjeti/protumačiti – koristan dio poruke/podatka. *Informacioni biti* u nekoj poruci su oni koji stvarno nose informaciju (koji su korisni).

Redundancija — Dupliranje, ponavljanje elemenata ili informacija; u kontekstu kodne teorije − dodavanje bitova poruci kako bi se omogućila detekcija i korekcija pogreške. Redundantni ili *biti parnosti* su oni koji se dodaju poruci kako bi se na osnovu njih moglo izvršiti ispravljanje poruke u uslovima kada se u prenosu dogode pogreške.

Signal — Signal je fizički koncept – veličina koja nosi informaciju (npr. namagnetisanje, napon, struja, optički signal, zvuk, grafički simboli, slike itd.).

Alfabet — Skup simbola iz kojih se generišu poruke, npr. *binarni* {0,1} ili skup slova našeg ili engleskog jezika itd. *Ulazni alfabet* je skup simbola koje treba kodirati, dok je *izlazni alfabet* skup simbola kojim se vrši kodiranje.

Izvor — Strana u sistemu za prenos informacija koja generiše poruke (sinonim: *predajnik*); alternativno alfabet – skup simbola – iz kojeg se generišu poruke.

Prijemnik — Strana u sistemu za prenos informacija koja prima i tumači poruke.

1 Približno navedeno po redosljedu pojavljivanja.

Diskretan — Odnosi se na sistem, promjenljivu, alfabet, a predstavlja ono što uzima vrijednosti iz konačnog (diskretnog) skupa.

Kontinualan — Odnosi se na sistem, promjenljivu ili alfabet sa beskonačnim brojem vrijednosti koje mogu biti u nekom domenu definisanosti (kod nas se može koristiti i sinonim *analogni*).

Koder izvora — Element sistema za prenos informacija koji se koristi za kompresiju poruke (uklanjanja redundancije) u cilju njenog kompaktnijeg zapisa (radi pojeftinjenja prenosnih i memorijskih resursa ...).

Koder kanala — Element sistema za prenos informacija koji je zadužen da omogući dekodiranje poruke na prijemnoj strani na osnovu redundancije koju ovaj blok dodaje u sistem (dodaje se dio bitova tako da se omogući dekodiranje poruke čak i u slučaju grešaka u kanalu).

Kanal — Medij za prenos informacija (spojni put, fizički medij koji povezuje izvor i prijemnik informacije, može da bude žica, optički kabal, bežični link, podvodna ultrazvučna komunikacija, ali može biti i papir, CD, magnetna traka, DVD, hard-disk itd.). Sinonimi – *komunikacioni kanal* ili *kanal za prenos informacija*.

Šum — Fizička pojava, smetnja, koja djeluje na korisni signal ili u komunikacionom kanalu koji se ne može opisati determinističkim zakonom, pa se za njen opis koriste probabilistički modeli (vjerovatnoće). Poznati tip šuma je *Gausov* ili šum s normalnom raspodjelom. Koristi se, između ostalog, za modelovanje kretanja u materiji, prouzrokovanog termičkim uticajima. Složeniji tipovi šuma su Rejlijev, Laplasov, Košijev itd. *Aditivni šum* je onaj koji je sabran sa pojavom, a *multiplikativni* je onaj koji je pomnožen sa signalom od interesa. *Signal-zavisni šum* mijenja statističke karakteristike kako se mijenja jačina signala.

Interliver — Blok koji permutuje ulaznu poruku, odnosno svi biti ulazne poruke se nalaze u izlaznoj, ali je njihov redosljed izmijenjen (često se koristi za poboljšanje performansi kodiranja). Suprotnu operaciju na strani prijema obavlja *deinterliver*.

Kriptografija — Podoblast teorije kodova koja se bavi prenosom tajnih informacija koje su nedostupne neovlašćenim korisnicima (od grčkih riječi *krypto* – tajno, *graphos* – pisati).

Deterministički Sistemi, signali, pojave imaju precizno definisane relacije, npr. kod sistema, na osnovu datih ulaznih, dobićemo tačno određene izlazne vrijednosti.

Binarni Alfabet koji se sastoji od dva simbola, *ternarni* od tri, *kvaternarni* od četiri itd.

Stohastički Sistemi, signali, pojave imaju neodređenost, odnosno moraju se modelovati putem slučajnih/probabilističkih modela, to jest na osnovu zakona teorije vjerovatnoće. Sinonim – *slučajan*.

Vjerovatnoća Vjerovatnoća je način kvantifikovanja kojim izražavamo šansu da se neka pojava dogodi. Varijante: *uslovna vjerovatnoća* – vjerovatnoća nekog događaja, pod uslovom da je poznat ishod nekog drugog događaja; *združena vjerovatnoća* – vjerovatnoća ishoda više događaja. Kada imamo više događaja, vjerovatnoća pojedinačnog se naziva *marginalnom*. Pod Bajesovom (engl. *Thomas Bayes*) vjerovatnoćom podrazumijevamo uslovnu vjerovatnoću kada znamo ishod, a želimo da odredimo vjerovatnoću uzroka.

Komplement U smislu vjerovatnoće je dopuna vjerovatnoće datog događaja do vjerovatnoće sigurnog događaja (1-p).

F-ja raspodjele Funkcija raspodjele i *funkcija gustine raspodjele* se obično koriste za opis slučajnih pojava koje mogu uzeti kontinualne vrijednosti (neprebrojivo mnogo vrijednosti), obično definisane nad nekim intervalom.

Očekivanje Srednja vrijednost neke slučajne promjenljive je osnovna statistika koja opisuje karakteristike slučajnih događaja. Druga bitna statistika je *varijansa*, koja je kvadrat *standardne devijacije* i koja predstavlja mjeru odstupanja slučajne promjenljive od srednje vrijednosti.

Estimacija Postupak kojim se procjenjuje neka veličina (kod nas, vjerovatnoća nekih događaja, ishoda ili simbola). Pojam se koristi i za vrijednost procjene.

Nezavisni Događaji kod kojih ne postoji uslovljenost, odnosno pojava jednog ne govori ništa niti utiče na pojavu drugog i obrnuto.

Slučajni proces Pojava čije izvršavanje, odnosno vjerovatnoća odvijanja zavisi i od vremena; dakle, ishod slučajnog događaja može se pretpostaviti kao jedna funkcija (a ne jedna vrijednost).

Ansambl Skup svih mogućih (vremenskih) funkcija koje odgovaraju nekom slučajnom procesu. Pojedinačni element ansambla naziva se *realizacijom*.

Stacionaran Slučajni procesi čija se funkcija gustine raspodjele ne mijenja tokom vremena (ovo se naziva i *stacionarnošću u užem smislu*). Stacionarnost u širem smislu podrazumijeva nepromjenljivost srednje vrijednosti i varijanse tokom trajanja signala.

Ergodičnost Slučajni proces je ergodičan ako su njegove statistike iste, računate po vremenu i za čitav ansambl (statističke karakteristike se mogu odrediti iz jednog dovoljno dugog posmatranja).

Grupa Algebarska struktura definisana nad skupom i odgovarajućom operacijom, koja zadovoljava osobine asocijativnosti, postojanja nultog (jediničnog) i inverznog elementa. Specijalni tip grupe je *Abelova* ili *komutativna grupa* kod koje važi i komutativnost. *Ciklične grupe* posjeduju element (*generator*) koji, kada se na njega primijeni operacija, daje sve (nenulte) elemente grupe.

Polje Algebarska struktura izvedena nad grupom u kojoj postoje dvije operacije. Pored osobina definisanih za Abelovu grupu, nad obje operacije (za množenje inverzija nije definisana za nulti element) važi i operacija distributivnosti.

Vekt. prostor Algebarska struktura (polje) definisana nad vektorima (ili polinomima) s odgovarajućim operacijama.

Polinom Izraz (funkcija) sa jednom ili više promjenljivih, sa operacijama sabiranja oduzimanja i množenja.

Prost Broj ili polinom je onaj koji se ne može prikazati kao proizvod manjih brojeva ili polinoma nižeg reda (osim jediničnog elementa) ili koji je djeljiv samo sa samim sobom i sa jedinicom. Prosti polinomi koji zadovoljavaju ovu osobinu ponekad se nazivaju *nesvodivim*, a za (dovoljnost) zadovoljenje osobine prostog polinoma uvode se dodatni kriterijumi.

Euklidski alg. Postupak za određivanje najvećeg zajedničkog djelioca dva broja ili polinoma. Postoje brojne ekstenzije ovog algoritma, koje se koriste za dekodiranje naprednih kodova za kodiranje kanala (detekciju i korekciju pogreški).

Generator Odnosi se na polje ili grupu. Element koji, kada se nad njim provede odgovarajuća operacija definisana u polju ili grupi, daje sve ostale elemente skupa. Slično, *generatorski polinom* ili *generatorska matrica* mogu da produkuje sve ostale elemente iz vektorskog prostora (sve moguće kodne riječi).

Korelacija Statistička veza između pojedinih događaja može se koristiti kao sinonim za zavisnost. Postoji matematička funkcija koja opisuje zavisnost slučajnih događaja/procesa. *Autokorelacija* je mjera zavisnosti pojedinačnog događaja, računatog u različitim trenucima.

Spektar Sinonim − *frekventni domen*. Često se neka pojava, umjesto u vremenski ili neki drugi adekvatni domen, transformiše u spektralni – frekventni domen jer je analiza ovakvih pojava ili procesa često jednostavnija, a ponekad i jedino moguća. *Spektralna gustina snage* je kvadrat modula Furijeove (fr. *Fourier*) transformacije slučajnog procesa ili autokorelacione funkcije.

Kongruencija Matematička operacija koja daje ostatak pri dijeljenju dva izraza. Kongruentno znači da dva izraza imaju isti ostatak pri dijeljenju sa nekim brojem ili funkcijom.

Entropija Mjera količine informacije (u fizici − mjera neuređenosti nekog sistema). Varijante: *uslovna entropija* (entropija jednog događaja kada je ishod drugog događaja poznat), *združena entropija* (združena neodređenost više događaja). *Binarna* ili *entropija binarnog događaja* je entropija koja se koristi za slučaj binarnog alfabeta. *Relativna entropija* ili *Kalbek−Lejblerova* (engl. *Solomon Kullback* i *Richard Leibler*) d*istanca* (ili divergencija) je mjera sličnosti dva slučajna događaja ili alfabeta.

Međus. infor. Količina informacije koju dijele dva slučajna događaja (alfabeta).

Lančano prav. Koristi se za opis veza među entropijama: entropija pojedinačnog događaja se smanjuje kako se povećava informacija o tom doga-

đaju, dobijena posredno preko drugih događaja. Postoji lančano pravilo za uslovne vjerovatnoće.

Markovljev Sistem ili lanac je metodologija grafičkog i matematičkog modeliranja i opisa sistema/alfabeta/događaja kod kojih u jednom trenutku postoji zavisnost dešavanja od dešavanja koja su im prethodila. Sistem *nultog* reda je onaj kod kojeg je svaki simbol/događaj nezavisan, *prvog* reda je situacija kada simbol zavisi od neposredno prethodnog simbola, *k-tog* reda opisuje zavisnost u odnosu na *k* prethodnih simbola. Markovljev sistem se naziva *ergodičnim* ili nesvodivim ako se iz bilo kog stanja može preći u drugo stanje u konačnom broju koraka. Stanja *komuniciraju* ako se između njih može napraviti prelaz u konačnom broju koraka. *Komunikaciona klasa* je podskup Markovljevog sistema gdje svi čvorovi međusobno komuniciraju. Komunikaciona klasa je *zatvorena* ako se ne može napustiti. Stanje se naziva *rekurentnim* (*esencijalnim*) kada se iz svakog stanja, u koje se može doći iz njega, može i vratiti u to stanje. Stanje je *periodično* kada se poslije određenog broja koraka (periode) možemo vratiti u dato stanje. *Tranzijentna* stanja su ona kod kojih postoji nenulta vjerovatnoća da se u njih ne vratimo nakon napuštanja. *Apsorbujuće* stanje se ne može napustiti.

Stacionarno U udžbeniku se pod stacionarnim stanjem podrazumijeva stanje kod kojeg se vjerovatnoće dalje ne mijenjaju – stabilno stanje.

Trelis Tip dijagrama koji se koristi za prikaz Markovljevih sistema, kao i za vizuelizaciju konvolucionih kodova itd.

Konveksnost Funkcija je konveksna (udubljena) u nekom intervalu ako je moguće povući pravu koja spaja dvije tačke te funkcije u datom intervalu tako da je uvijek sama funkcija ispod te prave, dok je *konkavna* (ispupčena) ako prava koja spaja tačke na intervalu funkcije uvijek leži ispod same funkcije. U prvom slučaju, drugi izvod funkcije je u posmatranom intervalu nenegativan, dok je u drugom slučaju drugi izvod nepozitivna veličina.

Detekcija Kod kodiranja kanala predstavlja mogućnost da se detektuje (otkrije) postojanje greške, ali ne i sposobnost njenog ispravljanja.

Korekcija Kod kodiranja kanala podrazumijeva mogućnost da se određeni broj grešaka, koje se pojave u komunikacionom kanalu, isprave.

AEP Asimptotska ekviparticiona osobina je osobina stohastičkih sistema koja kaže da se od svih mogućih realizacija stohastičkog skupa sekvence sa nekom zajedničkom karakteristikom najčešće dešavaju tzv. *tipične sekvence*, gdje se svaki simbol pojavljuje „očekivani“ broj puta.

Tipična Sekvenca predstavlja skup sekvenci čija je ukupna vjerovatnoća pojavljivanja bliska jedinici. Postojanje ovakve sekvence je posljedica zakona velikih brojeva i asimptotske ekviparticione osobine. *Visokovjerovatni skup* čine sekvence čija je ukupna vjerovatnoća bliska jedinici.

Konvergencija Za neku sekvencu (red) kažemo da konvergira po vjerovatnoći kada je moguće za svako ε, koje predstavlja apsolutnu razliku između vrijednosti reda i vrijednosti ka kojoj konvergira, odrediti dužinu reda *n* takvu da za duže redove imamo vjerovatnoću razlike koja konvergira nuli.

Kompresija Postupak kojim se originalna informacija sabija tako da produkuje najmanju moguću količinu podataka na nekom memorijskom mediju ili da zauzima komunikacioni kanal najkraće moguće vrijeme; suštinski, originalnu poruku zapisujemo sa što je moguće manje simbola (bita). Dvije osnovne tehnike kompresije su *kompresija bez gubitaka* (originalna informacija se može rekonstruisati bez izmjena – gubitaka) i *kompresija sa gubicima* (originalna informacija se rekonstruiše sa malim, ponekad jedva opservabilnim izmjenama).

Singularni Kodovi gdje se dva ili više simbola ulaznog alfabeta preslikavaju u istu kodnu riječ izlaznog alfabeta. Ovo je dozvoljeno samo kod kodova sa gubicima. Suprotno je *nesingularni kod*.

Dekodabilni Partikularno *jednoznačno dekodabilni kodovi* su oni kodovi kod kojih se za svaku kombinaciju ulaznih simbola produkuje kod koji se može jednoznačno dekodirati.

Prefiksni Ili *trenutni kodovi* su oni koji se mogu dekodirati u trenutku prijema posljednjeg simbola izlaznog alfabeta kojim je kodiran

simbol ulaznog alfabeta. Prefiksnim se nazivaju zato što nijedna kodna riječ nije prefiks druge ispravne kodne riječi. Pogodno se prikazuju kodnim stablom. Prefiks je početni dio neke sekvence, a *sufiksom* se naziva i završni ili nadodati dio neke sekvence.

Koma kodovi Specijalan slučaj trenutnih kodova kod kojih je posljednji simbol ili kombinacija simbola oznaka kraja kodne riječ, npr. 00, znači da se završava kodna riječ bez obzira šta je prethodilo ovoj kombinaciji.

Optimalni Naziva se još i *kompaktnim kodom* – kod ili postupak kodiranja koji daje najmanju moguću prosječnu dužinu kodne riječi među svim jednoznačno dekodabilnim kodovima.

Grejov kod Kod koji je dobio ime po Frenku Greju (engl. *Frank Gray*). Koristi se kao pomoćni u kodiranju izvora (pa i za druge namjene). Kodne riječi koje predstavljaju susjedne vrijednosti u nekom alfabetu kodiraju se tako da se razlikuju samo na jednom bitu.

RLE kod Kod s pokretnom dužinom (engl. *Run length encodding*) je pomoćni kod u kodiranju izvora kojim se uzastopno pojavljivanje simbola u poruci kodira parom: simbol − broj pojavljivanja simbola.

Diferencijalni Kodiranje kod kojeg se kodira razlika uzastopnih simbola, a ne sami simboli (primjenjuje se kod sporopromjenljivih izvora). Postoje razne varijante ovih kodova (kod nekih se kodira razlika blokova, a ne simbola).

Šenon−Fano Postupak kodiranja – algoritam za kodiranje izvora predložen od strane Šenona i Fanoa u cilju minimizacije prosječne dužine kodne riječi. Kod funkcioniše rekurzivnim dijeljenjem kodnih riječi u dva podskupa sa približno jednakim vjerovatnoćama dok se ne stigne do jednog simbola. Kod se pokazao neuspješnim zbog nejasnog i nepreciznog postupka dijeljenja u podskupove.

Rječnik Ili *kodna knjiga* (engl. *dictionary*) je spisak u kome se nalaze podaci o tome kojem simbolu ulaznog alfabeta odgovara koji simbol izlaznog alfabeta. Da bi se izbjeglo slanje informacija o rječniku sa predajnika prema prijemniku i kako bi se komunikacija rasteretila, razvijen je veliki broj alfabetova koje posjeduju

prijemnici i predajnici. Posebna klasa kodova *zasnovanih na rječniku* (engl. *dictionary based codes*) podrazumijeva formiranje rječnika i na prijemniku i na predajniku po unaprijed utvrđenim pravilima, bez prenošenja samog rječnika.

Hafmenov — Kod koji je razvio Dejvid Hafmen (engl. *David A. Huffman*), 1952. godine, koji pod datim okolnostima (poznatim vjerovatnoćama simbola alfabeta) daje najmanju prosječnu dužinu kodne riječi. Pripada klasi *entropijskih kodova* koji su dizajnirani da se primaknu predviđanjima I Šenonove teoreme (prosječnoj dužini kodne riječi koja je jednaka entropiji alfabeta koji se kodira).

LZ(W) kodovi — Klasa kodova zasnovanih na rječniku, koji su predloženi od Lempela, Ziva i Velča (engl. *Welch*). Do danas su ostali najbolji kodovi za kodiranje izvora bez gubitaka. Optimalni su pod uslovom nepoznavanja vjerovatnoća i uslovnih (združenih) vjerovatnoća simbola i sekvenci simbola.

Autoregresivni — Signal koji se s određenom (visokom) tačnošću može opisati kao težinska suma sopstvenih prethodnih odbiraka.

Memorija — Ima više značenja i u samoj knjizi. U kontekstu informacija, kanala i Markovljevih sistema ukazuje na uticaj prethodnih simbola na tekući. U kontekstu kodera, u pitanju su ćelije registra za smještaj djelova kodne riječi koji se koriste u procesu kodiranja. Kada se govori o *kanalu bez memorije*, jedan izlazni simbol zavisi od samo jednog ulaznog simbola u povorci i ne postoji zavisnost u odnosu na druge simbole. Kod *kanala sa memorijom*, zbog fizičkih pojava u kanalu ili konstrukcionih karakteristika prenosnog sistema, primljeni simbol zavisi od više simbola sa ulaza.

Registar — Memorijski blok koji se sastoji od ćelija u kojima se upisuju bitovi. Najčešće koristimo pomjerački registar koji ima takve ćelije da ulazni bit pomjera bit iz ćelije u narednu itd. Realizuje se obično putem sklopova koji se nazivaju flip-flopovi.

Kapacitet — Veličina koja karakteriše koliko kanal može prenijeti informacija za jedan simbol s ulaza.

Simetričnost Karakteristika kanala. *Strogosimetrični* kanali imaju matricu tranzicije kod koje su sve vrste permutacije prve vrste i sve kolone su permutacije prve kolone. Kod *slabosimetričnih kanala* vrste su permutacije prve vrste, dok su sume kolona konstantne. *Simetričnost po blokovima* podrazumijeva kanale kod kojih se kolone mogu grupisati u strogosimetrične matrice. Suprotno od prethodnog su *nesimetrični kanali*.

Propusni ops. Frekvencijsko područje raspoloživo za prenos informacija.

Iskorišćenost Dio kapaciteta kanala koji se koristi u datoj komunikaciji.

Kodna teorema Poznata i kao II Šenonova teorema. Uspostavlja vezu između kodnog odnosa i kapaciteta kanala, odnosno kaže da je moguće pod razmatranim uslovima dizajnirati kod sa kodnim odnosom koji je manji od kapaciteta kanala, a koji daje vjerovatnoću greške u procesu dekodiranja koja teži nuli.

Kodni odnos Odnos broja informacionih bita sa brojem bita u kodnoj riječi. Što je manji, dodata je veća redundancija u kodnu riječ radi mogućnosti ispravki grešaka koje se događaju u kanalu.

Blok kodovi Kodovi za kodiranje kanala namijenjeni za slučaj kada nam je potrebno ispravljanje greške. Osnovna karakteristika im je konstantna dužina kodne riječi. Rade tako što informacionu poruku podijele u blokove i svaki blok kodiraju fiksnom dužinom kodne riječi.

Pravougaoni Tip primitivnih blok kodova za kodiranje kanala. Sastoje se iz više poruka „naslaganih“ u tabelu, gdje krajnji red i krajnja kolona predstavljaju bite parnosti. Jedna očekivana greška nalazi se na presjeku indicirane kolone i indicirane vrste.

Trougaoni Jednostavna grupa blok kodova za kodiranje kanala, kod kojih se može zamisliti da je poruka postavljena trougaono. Ovdje su biti parnosti na hipotenuzi i svaki od njih kontroliše sopstvenu vrstu i kolonu, a greška se ponovo nalazi u presjeku dva indikovana bita parnosti.

Repetitivni Kodovi kod kojih se informacioni bit ponavlja više puta i odluka se donosi većinom glasova. Ponekad se nazivaju kodovima sa

majoritetnom (većinskom) logikom. Postoje slični postupci za dekodiranje *konvolucionih kodova*.

Hemingovi — Kodovi koji predstavljaju najkvalitetniju grupu postupaka za blok kodiranje s mogućnošću ispravke jedne pogreške. Dobili ime po Ričardu Hemingu (engl. *Richard W. Hamming*). Pod posmatranim uslovima predstavljaju optimalan kod (sa najmanje redundantnih bita). *Prošireni Hemingovi kodovi* imaju dodatnu provjeru parnosti za čitavu riječ, koja omogućava, pored ispravljanja jedne pogreške, i detekciju dvije.

Distanca — Hemingova distanca predstavlja broj pozicija na kojima se dvije kodne riječi razlikuju. Ponekad se naziva *rastojanjem* ili *mjerom*. *Minimalno Hemingovo rastojanje (distanca)* je minimalno rastojanje između dvije različite ispravne kodne riječi nekog koda. Na osnovu minimalnog Hemingovog rastojanja određujemo sposobnost koda da ispravlja ili detektuje pogreške.

Težina — *Hemingova težina* − broj nenultih pozicija u kodnoj riječi.

Binarna poz. — Pod pravilnom binarnom pozicijom podrazumijevaju se one pozicije u binarnoj riječi koje se mogu zapisati u obliku stepena broja 2 (2^0 = prva, 2^1 = druga, 2^2 = četvrta, 2^3 = osma itd.).

Sfera — Sfera je dio n-dimenzionog kodnog prostora takav da se sve riječi u njemu nalaze na rastojanju koje je manje od neke zadate vrijednosti (*radijusa* sfere) u odnosu na kodnu riječ koja se nalazi u centru sfere.

Zapremina — Broj kodnih riječi koje se nalaze u sferi (kardinalnost sfere).

Kontrolna mat. — Matrica dimenzija $(n-k)\times n = m \times \times n$, gdje je k broj informacionih bita, n ukupan broj bita u kodnoj riječi, dok je m broj redundantnih bita (provjera parnosti). Koristi se za detekciju i korekciju grešaka u kodnoj riječi. *Generatorska matrica*, dimenzija $k \times n$, koristi se u paru sa kontrolnom.

Sindrom — Funkcija (izražena preko polinoma ili matrice) koja služi da indicira da li je neka primljena poruka neispravna, pa čak i gdje se greška dogodila.

Kodna teorija — *Algebarska kodna teorija* je matematička oblast koja nudi sofisticirane alate za analizu, kreiranje, kodiranje i dekodiranje kodova za kodiranje kanala.

Povrat. sprega — Važan koncept u inženjerstvu. Na osnovu rezultata sa izlaza, može se vršiti korektivna akcija u smislu upravljanja sistemom. U kodnoj teoriji obično se odnosi na sklopove kod kojih izlaz istovremeno utiče i na stanje procesiranja narednih bitova.

Sistematski — Odnosi se na kodove za korekciju pogreški kod kojih su informacioni biti direktno, neizmijenjeno zastupljeni u kodnoj riječi.

Interliver — Sklop koji vrši permutaciju kodne riječi. Koristi se iz više razloga u postupcima kodiranja kanala, od izbjegavanja uticaja uzastopnih pogreški, do popravljanja kodnih karakteristika kod turbo-kodova. Inverznu operaciju obavlja sklop koji se naziva *deinterliver*.

Nesvodiv — Polinom koji nije djeljiv ni sa jednim *moničnim* polinomom (kojem je koeficijent uz najveći stepen jednak 1) nižeg reda osim sa jedinicom. Nazivaju se i *prostim*, ali se ponekad za proste polinome koji su pogodni za osnovu kodiranja uvode dodatni uslovi.

Ciklični — Pod cikličnim kodom podrazumijevamo takav kod ili kodni postupak kod kojeg se cikličnim pomjeranjem svake kodne riječi dobija ispravna kodna riječ.

Konjugati — Konsekutivni kvadratni stepeni nule prostog polinoma koji se, između ostalog, koriste da odrede stepene polinoma koji se pojavljuju kod BCH kodiranja.

Lokator — Polinom lokator pogreške nastaje kao međurezultat provođenja Euklidskog algoritma u procesu BCH kodiranja. Nule ovog polinoma određuju lokaciju pogreški u primljenoj riječi. Postoji i *polinom vrednovanja pogreške*, čije vrijednosti korespondiraju s težinom pogreški.

Konvolucioni — Tip kodova za kodiranje kanala koji su u jednom periodu bili najpopularniji i najzastupljeniji u ovoj oblasti. Suštinski, sastoje se od nekoliko blok kodova. Ulazna sekvenca se dijeli u duže

podnizove i kodira ovim kodom (dakle, podsekvence nisu strogo ograničene kao u slučaju blok kodova).

Turbo-kod Predstavljaju grupu kodova za kodiranje kanala koji su danas među najpopularnijim. Slični su konvolucionim kodovima, s tim da se koristi interliver u procesu kodiranja. U procesu dekodiranja vrši se tzv. meko odlučivanje, odnosno u upotrebi su primljene amplitude signala, a ne binarni brojevi.

Viterbi Algoritam za estimaciju skrivenih stanja u pojavama. Predstavlja varijantu algoritama za određivanje optimalne putanje. Razne ekstenzije ovog algoritma se intenzivno koriste u kodiranju, a posebno u dekodiranju kanala. Postupak je iterativan, gdje se kodni simbol u narednom trenutku određuje na osnovu mogućih kodnih simbola (odnosno putanja) iz prethodnog trenutka. Pamte se *parcijalne najbolje putanje* do svake tačke u kodnom stablu (trelisu).

Ogranič. duž. Pojam iz oblasti kodiranja konvolucionih kodova (engl. *constraint length*). Konvolucioni kodovi su suštinski dizajnirani da kontinualno vrše kodiranja bez podjele ulaznog toka podataka u podsekvence. Međutim, pokazano je da dobit takvog postupka saturira nakon određene dužine segmenta, pa je radi pojednostavljivanja računanja, memorijskih zahtjeva i procesa dekodiranja izvršeno ograničavanje dužine podsekvenci koje se kodiraju konvolucionim kodom.

Ortogonalan Osobina dva ili više objekata da im je skalarni proizvod jednak nuli. U teoriji kodova smatramo da je sindrom (ili kod) ortogonalan u odnosu na neki simbol kada svaki bit sindroma zavisi od datog bita primljene riječi.

Meki ulaz Umjesto da primljenu poruku tretiramo kao bite (dobijene nakon primjene poređenja sa pragom), u procesu dekodiranja koristimo mjerene vrijednosti.

Vjerodostojnost U engleskoj terminologiji *likelihood* je odnos uslovnih vjerovatnoća da li je neka pretpostavka zadovoljena u statističkom uzorku. U teoriji kodova pretpostavlja se da li je primljeni simbol 0 ili 1. Najčešće se definiše i koristi u logaritamskoj skali.

Rid−Mulerovi Kodovi (engl. *Reed−Muller*) za kodiranje kanala sa dobrim vezama sa diskretnim unitarnim transformacijama.

Golaj Tip *perfektnih* kodova za ispravljanje tri pogreške. Pod perfektnim podrazumijevamo kodove koji zadovoljavaju idealno pakovanje sfera.

Pakovanje Hemingova ili granica pakovanja sfera predstavlja odnos broja riječi koje se mogu konstruisati datim brojem bita podijeljeno sa brojem riječi koje prilikom dekodiranja pridružujemo jednoj ispravnoj kodnoj riječi. Kodovi koji dostižu ovu granicu sa jednakošću nazivaju se perfektnim.

Varšamova Granica kojom se pokušava dokazati postojanje koda sa datim parametrima (u literaturi poznata i kao *Gilbert−Varšamova granica*).

Gopa Tip perfektnih kodova (rus. Вале́рий Дени́сович Го́ппа).

Singleton Granica koja je dobila ime po Ričardu Kolom Singltonu (engl. *Richard Collom Singleton*) i predstavlja relativno grubu procjenu za veličinu proizvoljnog blok koda date dužine, veličine i minimalne distance.

Vandermond Matrica koja je dobila ime po slavnom francuskom matematičaru (muzičaru i hemičaru) Vandermondu (franc. *Alexandre-Théophile Vandermonde*). Posjeduje specijalnu strukturu da su redovi geometrijske progresije prethodnih redova. Determinanta ove matrice se lako računa.

Rid−Solomon Blok kodovi (engl. *Reed−Solomon*) razvijeni 60-ih godina prošlog vijeka nad proizvoljnim alfabetom (ne nužno binarnim). Bliski su BCH kodovima, posebno u smislu dekodiranja.

Tenerov graf Graf sa čvorovima koji predstavljaju provjere parnosti, odnosno bite u kodnoj riječi sa vezama među njima. Koristi se za opis LDPC kodova.

OZNAKE [2]

$\sum_{i=1}^{N}$	Suma niza po indeksima i koji se mijenjaju od 1 do N.
$f(x)$	Funkcija jedne promjenljive.
$f'(x), f''(x)$	Prvi i drugi izvod.
$df(x)/dx$	Izvod po promjenljivoj x.
$\partial / \partial x$	Parcijalni izvod – izvod funkcije više promjenljivih po promjenljivoj x.
$f'''(x), f^{(p)}(x)$	Treći i opšti p-ti izvod funkcije.
$a \to b$	a teži b.
∞	Beskonačno.
$[a_1, a_2, \ldots, a_n]$	Vektor (korišćene su i oznake s indeksiranjem od nule: $[a_0, a_2, \ldots, a_{n-1}]$, kao i $\{a_0, a_2, \ldots, a_{n-1}\}$, odnosno boldirano slovo **a**).
$\int$, $\int_a^b$	Neodređeni i određeni integral (u datim granicama).
log, $\log_b$	Logaritam, logaritam s osnovom b.
$\binom{n}{k}$	Broj permutacija, odnosno broj skupova od k elemenata koji se mogu generisati iz skupa od n elemenata (čita se n nad k).
ε, δ	Male vrijednosti (bliske 0) ili male vjerovatnoće.
$\delta(x)$	Dirakova (engl. *Paul Dirac*) funkcija različita od nule za $x = 0$.
$\|\ \|$, D, N	Kardinalnost (broj elemenata) skupa (alfabeta).
^{T}	Transponovana matrica.
max, $\max_x$	Maksimum i maksimum neke funkcije računat preko promjenljive x.
min, $\min_x$	Minimum i minimum neke funkcije računat preko promjenljive x.

[2] Spisak oznaka je formiran približno po redosljedu pojavljivanja, uz grupisanje srodnih. Naglašeno je kada se neka oznaka koristi u različitim kontekstima.

sign	Funkcija znaka daje rezultate 1, 0 i -1 za pozitivne, nulte i negativne argumente respektivno.
$\lceil\ \rceil, \lfloor\ \rfloor$	Za argument koji nije cijeli broj – zaokruživanje na veći i na manji cijeli broj.
$\oplus$	Ekskluzivno ili. Često se u istim kontekstu koristi oznaka $+$.
μ, E{}	Srednja vrijednost – matematičko očekivanje.
σ,σ^2	Standardna devijacija i varijansa šuma.
$\subset, \cup$	Podskup i unija skupova.
$\in$	Element skupa.
det[]	Determinanta matrice.
X,A	Alfabet, skup.
$\{x_1, x_2, \ldots, x_N\}$	Konačni skup sa N elemenata.
x_i	Element skupa – simbol alfabeta.
p, P, P_e, P_E	Vjerovatnoća, vjerovatnoća greške.
p_i	Vjerovatnoća pojavljivanja i-tog elementa iz skupa.
$\hat{p}$	Procjena (vjerovatnoće).
$p(\mathbf{a})$	Vjerovatnoće simbola nekog alfabeta prikazane u vektorskom obliku (**a** je vektor simbola datog alfabeta).
p	Vektor vjerovatnoća pojave simbola alfabeta. Koristi se kod Markovljevih sistema.
$X \to Y \to Z$	Označava Markovljev lanac, odnosno niz slučajnih procesa koji redom zavise jedan od drugog.
k_i	Broj pojavljivanja nekog (i-tog) elementa u skupu.
A,B	Događaji.
AB	Združeni događaj (dogodilo se i A i B).
$A\|B$	Uslovni događaj (dogodilo se A pod uslovom da se B dogodilo).

$P(A)$	Vjerovatnoća događaja (ishoda, simbola).
Pr()	Povremeno se vjerovatnoća neke pojave ili događaja označava ovako da bi se jasnije ukazalo da je funkcionalna zavisnost probabilistička. Uglavnom se ovakva oznaka koristi u nekim teorijskim izvođenjima (Pr skraćeno od engl. *probability*).
q, Q	Povremeno se koristi kao komplement vjerovatnoće greške ($q = 1\text{-}p$) ili kod kodova za detekciju pogreške kao vjerovatnoća detekcije pogreške.
$P(AB)$, $P(A\|B)$	Vjerovatnoće združenog i uslovnog događaja.
$P(x_1,x_2,...,x_n)$	Vjerovatnoća niza simbola (više događaja).
$P(\xi)$, $p(\xi)$	Funkcija raspodjele i funkcija gustine raspodjele kontinualne slučajne promjenljive.
$\lambda^{(n)}$	Vjerovatnoća greške u dekodiranju koda dužine n (koristi se kod iskaza kodne teoreme).
$p(\xi,\zeta)$,$P(\xi,\zeta)$	Funkcija gustine raspodjele i funkcija raspodjele združenih (dvodimenzionalnih) kontinualnih slučajnih događaja.
ν_i, $\nu(n)$	Šum – slučajni proces (korišćeno u kontekstu Gausovog šuma).
$\mu_X(t)$, $\sigma_X^2(t)$	Srednja vrijednost i varijansa slučajnog procesa računata po vremenu.
$\mu(t)$, $\sigma^2(t)$	Srednja vrijednost i varijansa slučajnog procesa računata po ansamblu.
$r_{XX}(t,\tau)$	Autokorelaciona funkcija za analogne procese, alternativno za diskretne procese oznaka je $r_{XX}(n,m)$.
$R_{XX}(\tau)$	Autokorelaciona funkcija za stacionarne procese kada zavisi samo od rastojanja među trenucima u kojima se posmatra proces.
$\Xi_T(\omega)$	Furijeova (fr. *Fourier*) transformacija slučajnog procesa računata na osnovu signala u intervalu dužine T.
$S_{XX}(\omega)$	Spektralna gustina snage slučajnog procesa.
P_{XX}	Snaga slučajnog procesa.

rem, mod, %	Ostatak pri dijeljenju.
r_k, q_k, u_k, v_k	Vrijednosti koje se javljaju u k-tom koraku provođenja Euklidskog algoritma (r_k – ostaci, q_k – količnici, u_k i v_k – pomoćne vrijednosti).
u(x)	Hevisajdova odskočna funkcija (jednaka 1 za $x > 0$ i nula drugdje).
ρ, ξ, η	Slučajne promjenljive.
$H(X)$	Entropija događaja (skupa, alfabeta).
$H_b(X)$	Entropija računata sa logaritmom za osnovu b.
$H(p)$	Entropija binarnog događaja sa vjerovatnoćom jednog od simbola p.
$H(X,Y)$	Entropija združenog događaja. Združena entropija više događaja se označava sa $H(X_1, X_2, ..., X_n)$.
$H(Y\|X)$	Entropija uslovnog događaja (entropija Y pod uslovom da se dogodilo X). Sinonim − uslovna entropija. Uslovna entropija u odnosu na više događaja označava se kao: $H(Y \mid X_1, X_2, ..., X_n)$.
$H(Y\|X=x)$	Entropija uslovnog događaja Y, pod uslovom da znamo da je ishod događaja $X=x$.
$H(\mathbf{r})$	Entropija vektora vjerovatnoća $\mathbf{r}$.
$H(x_1, x_2, ..., x_n)$	Entropija sekvence simbola x_1, x_2, ..., x_n.
$D(p \,\|\| \, q)$	Relativna entropija dvije raspodjele slučajnih promjenljivih $p(x)$ i $q(x)$. Sinonimi − *diskriminaciona funkcija* i *Kalbek–Lejblerova* (engl. *Kullback–Leibler*) *distanca*.
$I(X;Y)$	Međusobna informacija.
$I(X,Y;Z)$	Uslovna međusobna informacija događaja X i Y, pod uslovom da je poznat događaj Z.
$A_\varepsilon^{(n)}$	Tipični skup (skup tipičnih sekvenci sa n elemenata).
$B_\delta^{(n)}$	Visokovjerovatni skup sekvenci sa n elemenata.
λ	Lagranžev množilac, koristi se prilikom određivanja minimuma i maksimuma funkcije uz neko ograničenje.

$\bar{L}, \bar{L}_n$	Prosječna dužina kodne riječi i prosječna dužina kodne riječi kada se odjednom kodira više (n) simbola.
l_i, $l(x_i)$, $l_{\max}$	Dužina kodne riječi (za simbol indeksiran sa i ili simbol x_i) i maksimalna dužina kodne riječi.
L^*	Optimalna (minimalna dostižna) dužina jednoznačno dekodabilnog koda.
u, v, p	Parametri podintervala kod aritmetičkog koda.
$C(x_i)$	Kod kojim se kodira simbol x_i.
C_n	U kontekstu LZ(W) kodova, broj fraza u rječniku.
#	Terminacioni karakter – posljednji karakter u nekoj poruci. Oznaka korišćena kod aritmetičkih kodova.
F_a	Broj pojavljivanja simbola a u poruci (korišćeno kod aritmetičkog kodiranja).
P	Matrica tranzicije (matrica uslovnih vjerovatnoća) u kontekstu opisivanja i karakterizacije kanala. Radi kompaktnosti, uslovne vjerovatnoće, koje su elementi ove matrice, povremeno označavamo sa P_{ij}. Dio kontrolne i generatorske matrice (minor) kod blok kodova sa bitima parnosti grupisanim na krajevima kodne riječi (u kontekstu kodova kanala). Kod blok kodova za ispravljanje pogreški, oznaka se koristi za minor kontrolne, odnosno generatorske matrice (ponekad se minor označava i sa **R**).
C	Kapacitet kanala.
$\mathbf{x}^n(j)$	Poslata poruka dužine n za simbol j na strani predaje.
$\mathbf{y}^n(j)$	Primljena poruka dužine n za simbol j na strani prijema.
α	Korišćeno u kontekstu vjerovatnoće brisanja.
$Q(x)$, $\Phi(x)$	Funkcija greške i komplementarna funkcija greške. U literaturi se često označavaju kao erf(x) i erfc(x).
$I_0(\xi)$	Modifikovana Beselova funkcija prve vrste nultog reda.
P	Snaga signala.
R	Kodni odnos.

R_L	Kodni odnos konvolucionog koda sa ograničenjem (kod konvolucionih kodova).
n, k	Broj bita u kodnoj riječi, broj informacionih bita.
m	Broj redundantnih bita (provjera parnosti) $m = n\text{-}k$.
i_j	Informacioni biti (j je indeks pozicije bita u kodnoj riječi).
c_j	Biti u kodnoj riječi indeksirani sa j.
w_j	Težina simbola u provjerama kodne riječi (korišćena kod nebinarnih kodova).
(n,k)	Uobičajeni način označavanja blok kodova.
$(n,k)_d$	Blok kod sa minimalnom distancom d.
(n,w)	BCH kod, dužine n bita, koji može da ispravi w pogreški u kodnoj riječi.
M	Broj ispravnih kodnih riječi posmatranog koda (kod definicije kodnog odnosa).
c	Vektor koji predstavlja kodnu riječ.
i,**i**(x)	Vektor koji predstavlja informacione bite; odgovarajući polinom.
e,**e**(x)	Vektor koji predstavlja greške u kodnoj riječi; odgovarajući polinom.
r, **r**(x)	Vektor koji predstavlja primljenu kodnu riječ (sa eventualnim greškama); odgovarajući polinom.
p(x), **g**(x)	Prosti i generatorski polinom (kod nekih kodnih postupaka ovo su sinonimi).
$\mathbf{p}_i(x)$	Faktori generatorskog polinoma kod BCH koda.
$\mathbf{g}_i(x)$	Generatorski polinomi kod konvolucionih kodova.
$\mathbf{c}_i(x)$	Kodni polinomi kod konvolucionih kodova.
$\mathbf{h}_i$, $\mathbf{h}(x)$	Kolone kontrolne matrice koda, odnosno kontrolni polinom kod Rid−Solomonovih kodova.
a,α	Nula generatorskog/prostog polinoma.

$\mathbf{H}$	Kontrolna matrica koda.
$\mathbf{G}$	Generatorska matrica koda.
$\mathbf{g}(x)$	Kolone ili vrste generatorske matrice.
$\mathbf{S}$	Sindrom (ukazuje na „neispravnosti“ u primljenoj kodnoj riječi).
$\mathbf{I}$, I_k	Jedinična matrica dimenzija $k \times k$.
$\mathbf{S}(x)$, S_j	Sindromski polinom i koeficijenti sindromskog polinoma kod BCH koda.
$l(x)$, $u(x)$	Polinomi lokator pogreške i vrednovanja pogreške kod BCH kodiranja.
deg()	Stepen polinoma kod blok i konvolucionih kodova.
M,N	U kontekstu konvolucionih kodova korišćeno da označi parametre koda (maksimalni stepen – memoriju) i dubinu koda (broj bita ulaznog signala od kojeg zavisi dati bit izlaznog signala).
$\mathbf{G}_L$	Generatorska matrica konvolucionog koda s ograničenjem.
L	U kontekstu konvolucionih kodova ograničenje dužine koda (broj bita informacije nakon kojih se ponovo ponavlja postupak kodiranja od početka).
d	Distanca između kodnih riječi, minimalna distanca (zavisno od konteksta).
$d_H(\mathbf{x},\mathbf{y})$, $w_H(\mathbf{x})$	Hemingova distanca (broj pozicija na kojima se dvije kodne riječi razlikuju) i Hemingova težina (broj nenultih simbola u kodnoj riječi).
$\mathbf{q}(x)$, $\mathbf{o}(x)$	Količnik i ostatak pri dijeljenju polinoma.
$\tilde{\mathbf{q}}(x),\tilde{\mathbf{o}}(x)$	Količnik i ostatak pri dijeljenju polinoma u uslovima pojave pogreške u kanalu.
RM_p	Generatorska matrica Rid−Mulerovog koda.
$\mathbf{H}_P$	Matrica Adamardove transformacije/koda dimenzija $P \times P$, gdje je $P = 2^p$.

$\mathbf{F}_q^n$	Konačno polje sa q simbola alfabeta kojeg čine vektori dužine n (ili polinomi $n-1$ stepena). Alternativno, naziva se polje Galoa (fr. *Galois*) i može se označiti kao GF(q^n).
$L(x)$, $L(d\|x)$	Logaritamski odnos vjerovatnoća i uslovnih vjerovatnoća (odnos vjerodostojnosti).

SKRAĆENICE [3]

AAC	Napredno kodiranje audio (zvučnih) podataka (engl. *Advanced Audio Coding*), jedan od standarda za kompresiju zvučnih signala sa gubicima.
AEP	Asimptotska ekviparticiona osobina (engl. *property*) koja ukazuje (između ostalog) na mogućnost kodiranja izvora.
ARQ	*Automatic Repeat ReQuest* sistem prenosa kod kojega je, na osnovu koda za detekciju pogreške, prijemnik u stanju da detektuje postojanje greške u prenosu i da automatski zahtijeva (bez kontrole operatera) od predajnika ponovno slanje.
ASCII	*American Standard Code for Information Interchange* (Američki standardni kod za razmjenu informacija) propisuje 8 bita za svaki karakter iz osnovnog skupa karaktera na računaru. Jedan bit se može koristiti kao bit parnosti radi dobijanja informacije o grešci.
BCD	Binarno kodirani dekadni brojevi, svaki dekadni broj se kodira četvorobitnom sekvencom koja predstavlja binarni ekvivalent datog broja.
BCH	*Bose−Chaudhuri−Hocquenghem* kodovi imaju sposobnost ispravljanja više od jedne pogreške; pripadaju grupi blok kodova.
ber	Vjerovatnoća greške po bitu (engl. *bit error rate*).
bit	Mjerna veličina za količinu informacije (jedna jedinica ili nula). Ovo je mjera međunarodnog sistema (SI) mjera.
BPSK	Tip modulacije u digitalnim komunikacijama (engl. *Binary Phase Shift Keying*).

3 Navedene po abecednom redu.

BSC	Binarni simetrični kanal (*channel*) – kanal za prenos binarnih informacija (i na prijemu i na predaji se nalaze biti 0 i 1) sa jednakim vjerovatnoćama pogreške i kod slanja nule i kod slanja jedinice.
CBC	Jedan od načina (modova) funkcionisanja DES standarda za kriptografiju. Skraćenica od engl. *Cipher Block Chaining*.
CCITT	Konsultativni komitet za međunarodne telefonske i telegrafske komunikcije (engl. *Consultative Committee for International Telephony and Telegraphy*), ovdje navođen kao izvor pojedinih komunikacionih i kodnih standarda.
CD	*Compact disk* – kompaktni disk (tip optičkog diska za smještaj velike količine podataka).
CDMA	Digitalna modulacija kod koje se različiti kanali (signali koji se prenose) kodiraju različitim kodovima (engl. *Code Division Multiple Access*). Prilikom kodiranja, kod ove modulacije koriste se neki od učenih kodnih postupaka.
CR	Kompresioni odnos ili stepen (engl. *Compression ratio*) predstavlja količnik potrebne memorije za smještaj nekomprimovane sa memorijom potrebnom za smještaj komprimovane poruke.
CRC	Kodovi sa cikličnom provjerom redundancije (engl. *Cyclic Redundancy Check*) namijenjeni detekciji većeg broja pogrešaka u kodnoj riječi.
DES	Digitalni standard kriptovanja (engl. *Digital Encryption Standard*). Jedan od prvih kriptografskih standarda. Danas prevaziđen.
dit	Mjerna veličina za količinu informacija (dobija se korišćenjem dekadnog logaritma u izrazima za entropiju, međusobnu informaciju itd.).
DPCM	Diferencijalna impulsna kodna modulacija (modulator). Jedan od postupaka (uređaja) na strani predaje u komunikacijama, namijenjen za prenos diskretnih (digitalnih) informacija. Skraćenica od *Differential Pulse Code Modulation*.
DVB-T	Standard komunikacije za digitalni prenos videa (televizije) koji uključuje niz naprednih kodova i za kodiranje izvora i za kodiranje kanala.
DVD	*Digital Versatile Disc* (tip optičkog diska namijenjen za smještaj velike količine, prije svega, multimedijalnih podataka).

ECB	Jedan od načina (modova) funkcionisanja DES standarda za kriptografiju. Skraćenica od engl. *Electronic Codebook*.
FGK	*Faller−Gallager−Knuth* algoritam za dinamičko Hafmenovo kodiranje.
gcd	Najveći zajednički djelilac (engl. *Greatest Common Divisor*) dva cijela broja ili dva polinoma. Koristili smo i oznaku nzd.
IEEE	Institut inženjera elektrotehnike i elektronike (eng. *Institute of Electrical and Electronic Engineers*). Najveće svjetsko profesionalno udruženje koje je kreator brojnih standarda u komunikacijama i informatici.
ISBN	*International Standard Book Number* − broj (kod) koji na jedinstven način identifikuje knjigu. Ranije je imao deset simbola, a danas verzije (ponekad se nazivaju ISBN-13) imaju 13 simbola. Posljednji simbol je kontrolni.
JMBG	Jedinstveni matični broj građana. Prevaziđeni sistem kodiranja građana koji postoji u Crnoj Gori i nekim zemljama nastalim na prostoru bivše Jugoslavije. Jedinstveno određuje građanina podacima o datumu, mjestu rođenja, polu i redosljedu rođenja. Trinaesta cifra koda je kontrolna i služi za detekciju pogreški.
JPG, JPEG	Format digitalne slike s visokim stepenom kompresije sa gubicima (skraćenica od engl. *Joint Photographic Experts Group*).
LAN	Lokalna kompjuterska mreža (engl. *Local Array Network*).
LDPC	Kodovi „male gustine“ sa provjerom parnosti (engl. *Low-density parity-check*).
LLR	Logaritam odnosa vjerodostojnosti (engl. *Logarithm of Likelihood Ratio*).
LTE	Telekomunikacioni standard za bežične širokopojasne komunikacije, skraćenica od engl. *Long Term Evolution*.
LZ	*Lempel−Ziv* kodovi za kodiranje izvora (varijante LZ77, LZ78).
LZW	*Lempel−Ziv−Welch* kod za kodiranje izvora.
MDS	Kodovi koji dostižu Singletonovu granicu sa jednakošću (engl. *Maximum Distance Separable*).

NASA	*National Aero-Space Administration* − Nacionalna vazduhoplovna i svemirska administracija SAD – agencija koja je bila motor razvoja brojnih kodnih standarda.
nat	Mjerna veličina za količinu informacija (dobija se korišćenjem prirodnog logaritma u izrazima za entropiju, međusobnu informaciju itd.).
OFDM	Ortogonalna modulacija sa frekvencijskom podjelom − savremeni postupak prenosa više komunikacija preko jednog kanala, uz primjenu većeg broja frekvencijskih podnosilaca. Skraćenica od engl. *Orthogonal Frequency Division Multiplexing*.
QAM	Kvadraturna amplitudska modulacija (engl. *Quadrature Amplitude Modulation*).
QPSK	Tip modulacije u digitalnim komunikacijama (engl. *Quadrature Phase Shift Keying*).
READ	Generalizacija RLE koda namijenjena za slanje faksova. Neka vrsta dvodimenzionog RLE koda s referencom na prethodne redove slike (dokumenta). Skraćenica od engl. *Relative Element Address Designate*.
RLE	*Run length-encoding* kod kojim se kodira uzastopno pojavljivanje simbola u poruci parom: simbol − broj uzastopnih pojavljivanja simbola.
RM	Rid−Mulerovi kodovi (engl. *Reed−Muller*). Karakterišu se sposobnošću ispravke velikog broja pogreški.
RS	Rid−Solomonovi kodovi – varijanta BCH koda koja se uobičajeno koristi kod nebinarnih alfabeta specijalne strukture.
sign	Funkcija znaka (za argument veći od nule izlaz je jedan, za nulu izlaz je nula, dok je za negativne argumente rezultat −1).
SISO	*Soft-In Soft-Out* je tip dekodera koji se koristi kod turbo-kodova, ali i kod nekih drugih tehnika dekodiranja, kao što su, na primjer, konvolucioni kodovi. Odluka se donosi na osnovu aktuelnih mjerenja fizičke veličine na ulazu u dekoder a ne na osnovu binarnih predstava.
SOVA	Viterbijev algoritam sa „mekom“ odlukom (engl. *Soft Output Viterbi Algorithm*).

SS	Sačuvani prostor (engl. *Space Saving*) predstavlja procenat memorije koji je sačuvan (oslobođen za druge potrebe) u procesu komprimovanja (kodiranja izvora).
TIFF	Visokokvalitetni format zapisa digitalne slike, ranije bez gubitaka, a danas može da bude i sa gubicima. Skraćenica od *Tagged Image File Format*.
UMTS	Univerzalni mobilni telekomunikacioni sistemi – prevashodno evropsko rješenje u telekomunikacijama, standard koji koristi turbo-kodove.
USB	Žargonski se koristi za tip prenosivog diska, a, u stvari, naziv je za priključni (univerzalni serijski) port preko kojeg se može pristupiti računaru (engl. *Universal Serial Bus*).
VLSI	*Very Large Scale Integration* – tip integralnih kola (sistema) sa veoma visokim stepenom integracije (pakovanja komponenti).
VM	Vandermondova matrica.
WLAN	*Wireless Local Area Network* − bežična lokalna mreža s relativno sofisticiranim kodovima i postupcima radi korekcije grešaka u kanalu; koristi se svakodnevno na fiksnim i mobilnim uređajima u kućama, poslovnim objektima i javnim površinama.

ISBN 978-86-7664-236-6

Igor Đurović

TEORIJA INFORMACIJA I KODOVA